Título: *Integración de un ERP con una Tienda online*

Alumno: *Sandra Rodríguez Gómez*

Director/Ponente: *Antonio Cañabate Carmona*

Departamento: *Organización de Empresas (OE)*

Fecha: *21 de junio de 2010*

DATOS DEL PROYECTO

Título del Proyecto: *Integración de un ERP con una tienda online.*

Nombre del estudiante: *Sandra Rodríguez Gómez*

Titulación: *Ingeniería técnica en Informática de sistemas*

Créditos: *22,5*

Director/Ponente: *Antonio Cañabate Carmona*

Departamento: *Organización de Empresas*

MIEMBROS DEL TRIBUNAL *(nombre y firma)*

Presidente*: Ferrán Sabaté Garriga*

Vocal: *Robert Lukas Mario* Nieuwenhuis

Secretario: *Antonio Cañabate Carmona*

CUALIFICACIÓN

Cualificación numérica:

Cualificación descriptiva:

Fecha:

Índice

1. Introducción

1.1 Visión del proyecto

1.1.1 Presentación

Cada vez son más las empresas que optan por instalar un software empresarial para gestionar sus procesos de negocio. Entre los diferentes tipos de Software que podemos encontrar en el mercado, destaca, sin duda alguna, SAP. Este liderazgo se constata en el gran número de clientes que gestionan su información con dicho software.

Business and Corporate Overview (U.S. GAAP)	
Total Revenue	€10,671 million (FY 2009)
Software and Software-Related Service Revenue	€8,197 million (FY 2009)
Operating Income	€2,640 million (FY 2009)
Employees	47,578 (FTE at Dec. 31, 2009)
Customers	More than 95,000 in over 120 countries
Partners	More than 2,400 certified partners
R & D Investment	€1,608 million / 15% of total revenue (FY 2009)

Tabla 1

SAP es un sistema de planificación de recursos empresariales. Estos sistemas de información integran y manejan muchos de los negocios asociados a las operaciones de producción y a los aspectos de distribución de una compañía comprometida en la producción de bienes o servicios.

1.1.2 Perspectiva del producto

En general, por comercio electrónico se entiende toda compra realizada a través de Internet, cualquiera que sea el medio de pago utilizado. La característica básica del comercio electrónico reside en la orden de compraventa, la cual tiene que realizarse a

través de algún medio electrónico, con independencia del mecanismo de pago efectivo. El comercio electrónico entre empresas es una utilidad más que aporta Internet y que ha experimentado un gran auge en los últimos años.

El comercio electrónico realizado entre empresas es llamado en inglés *Business-to-business* o B2B. El B2B puede estar abierto a cualquiera que esté interesado o estar limitado a participantes específicos pre-calificados. El uso empresarial de la Web reduce errores, tiempo y sobrecostes en el tratamiento de la información.

En el cuarto trimestre de 2009, los ingresos del comercio electrónico en España alcanzaron los 1.574,2 millones de euros, con un aumento interanual del 26,1%. Este aumento representa una tendencia al alza como se puede observar en el Gráfico 1.

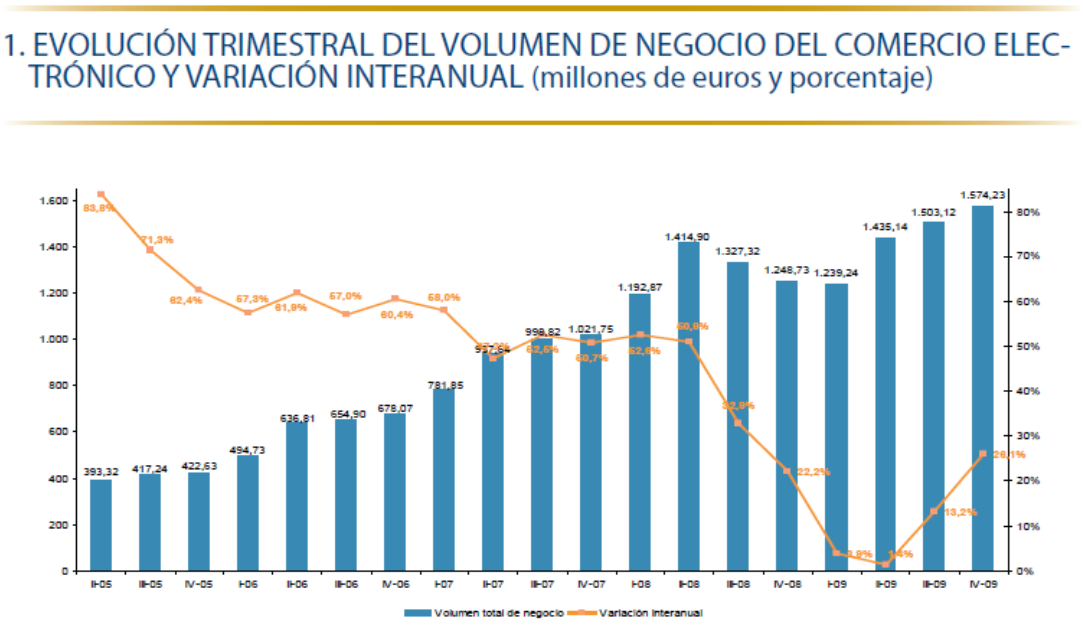


Gráfico 1: Informe el comercio electrónico 2009(C.M.T.)

Aproximadamente el 80% de las transacciones electrónicas son de este tipo, y la mayoría de los expertos predice que el B2B continuará creciendo

La transmisión de información entre un fabricante y el distribuidor de un producto o la transmisión entre el distribuidor y el comercio minorista son algunas de las relaciones que se pueden a establecer en este aplicativo. Este tipo de transmisión de información

se establece en un entorno B2B dónde el cliente no es ocasional, sino que se va establecer una relación de negocio con él de tal forma que va a marcar en gran medida algunos de los requisitos del sistema.

Uno de los procesos de negocio que es necesario en cualquier empresa que quiera vender sus productos es el de **ventas y distribuciones**.

El primer paso que debe dar el departamento de ventas es hacer que el cliente conozca sus productos mediante un sistema de pre-ventas. Cuando el cliente decide que un producto le interesa se pone en contacto con la empresa donde detalla los productos que necesita; lo que se conoce como pedido.

Los distintos métodos que actualmente las empresas utilizan para comunicar sus pedidos dependen en gran medida del volumen de pedidos que realizan.

Las empresas con un mayor volumen de pedidos utilizan el teleproceso de datos EDI (Electronic Data Interchange), el cual describe el intercambio electrónico de datos comerciales estructurados entre sistemas de distintas empresas.

Las pequeñas o medianas empresas pueden gestionar sus pedidos con un centro de llamadas o con los tradicionales métodos de preparación y envío de documentos a través de mensajería o email. Los clientes llaman o envían emails y la operadora entra el pedido de forma manual en el sistema.

Pero cada vez son más las empresas que utilizan la Web para informar a sus clientes sobre la compañía, aparte de sus productos o servicios. Esto facilita las relaciones comerciales, así como el soporte al cliente, ya que al estar disponible las 24 horas del día, las empresas pueden fidelizar a sus clientes.

El aplicativo Web pretende externalizar procesos que se ejecutan en SAP para garantizar una serie de beneficios a ambas partes en una empresa pequeña o mediana. De esta forma este tipo de empresa podría dejar de usar los métodos tradicionales de captación de pedidos y automatizar en un grado menor comparado con el teleproceso

de datos EDI el tratamiento de la información relacionada con los pedidos de sus clientes.

1.1.3 Beneficios para ambas partes

La externalización de procesos que se ejecutan en SAP aporta una serie de beneficios económicos que se describen a continuación.

1.1.3.1 Beneficios para la empresa

Cualquier usuario que quiera ejecutar procesos de negocios de SAP necesitará tener la parte cliente del software SAP instalada en la máquina desde la cual va iniciar la conexión a SAP. La opción online permite ahorrar las instalaciones de la parte cliente del software en las diferentes máquinas que utilizan los usuarios, disminuyendo el mantenimiento de esta parte del software.

Otro punto importante a tener en cuenta es la gestión de los usuarios en SAP. En un sistema SAP existen dos tipos de usuarios los ilimitados y los limitados. Los ilimitados pueden acceder a todas los recursos que proporciona SAP y los limitados sólo ejecutan determinados procesos.

El aplicativo Web permite que los usuarios de ventas sean capaces de generar altas de pedidos y clientes, lo que significa que actuarán como usuarios limitados sin necesidad de crear usuarios en el sistema SAP disminuyendo el número de usuarios que utilizan el sistema SAP.

De esta forma, se evita la costosa tarea de tener que crear y mantener usuarios en SAP para aquellos empleados que sólo utilizan procesos específicos como alta de pedidos y clientes en SAP.

Otro beneficio derivado está relacionado con las condiciones contractuales sobre la implantación del software SAP. Cualquier empresa contrata un límite de usuarios que utilizarán el sistema productivo. Cada usuario tiene un coste económico de forma que

al reducir el número de usuarios que utilizan SAP también reducimos el número de licencias contratadas, ya que la licencia se asigna al usuario.

Otro beneficio derivado es la disminución de la carga de trabajo para los empleados de ventas, ya que dotamos al cliente de capacidad para entrar sus pedidos en SAP vía online.

1.1.3.2 Beneficios para el usuario cliente

El principal beneficio es dar facilidad al cliente para gestionar sus pedidos. Los clientes pueden visualizar todos los pedidos que tienen pendientes así como, sus fechas de entrega, sus cantidades e información específica sobre ellos fácilmente desde cualquier lugar con conexión a internet. Además, los clientes tienen acceso a crear sus propios pedidos si lo consideran oportuno sin necesidad de intermediarios, con lo que se traduce en un aumento de su capacidad de autogestión.

1.2 Objetivos

Éste aplicativo pretende ser una vía de comunicación online entre SAP y el usuario, de forma que el usuario puede gestionar pedidos y crear nuevos clientes desde la Web y éstos se materializan en SAP de forma transparente a éste.
En ningún caso el aplicativo Web pretende replicar procesos ni estructuras de datos, el objetivo ha sido utilizar los recursos y datos que proporciona SAP.
El objetivo general del proyecto ha sido la construcción de una aplicación Web que permite utilizar algunos procesos de negocio soportados por SAP de forma online.
Los procesos de negocio que se ejecutan en SAP que puede utilizar el usuario desde el aplicativo Web son los siguientes:

- Generar el alta de un nuevo cliente
- Crear pedidos de tipo estándar.
- Consultar los pedidos pendientes de entregar.

1.3. Metodología.

Cuando se tiene que desarrollar un software de debe escoger una metodología, una estructura impuesta para seguir que marcará las etapas por las que deberá pasar un producto de software. Todo desarrollo de software implica un riesgo difícil de controlar por lo que una buena metodología puede ayudarnos a superar o a predecir posibles defectos en el ciclo de vida del software. Actualmente hay varios modelos para tales procesos. El modelo más conocido y antiguo es **el modelo en cascada** en el que se siguen unos pasos en orden secuencial, de esta forma no se empieza con el paso siguiente si el previo no está finalizado. Las etapas por la que debe pasar el desarrollo del software se detallan a continuación:

- Especificación de requerimientos
- Diseño
- Construcción
- Integración
- Pruebas
- Instalación
- Mantenimiento

Como se podrá comprobar una vez leído este proyecto, no se ha seguido estrictamente esta metodología porque el diseño se realizó antes que la especificación de requisitos. Sin embargo, las demás fases si se han realizado en orden expuesto en el modelo. Obviamente por no ser un proyecto real quedan excluidas las etapas de instalación y mantenimiento.

El proceso iterativo es otro de los principales modelos a tener en cuenta, consiste en asignar un tiempo determinado e ir mejorando y revisando las distintas partes del sistema. Uno de los más importantes es el Rational Unified Process (RUP) el cual propone un sistema en él que se van sucediendo distintas iteraciones pre-establecidas. Cada iteración supone nuevas funcionalidades en el sistema, nueva documentación y avance del proyecto. Tampoco se ha seguido en su totalidad este modelo, aunque sí ha sido útil en la implementación ya que se ha divido el sistema en entregables asociados

cada uno de ellos a una funcionalidad a implementar de esta forma se ha podido ver el avance del desarrollo del software.

1.3.1 Metodología utilizada.

Además de una metodología en cualquier proyecto es necesaria una comunicación entre los miembros del grupo. En el desarrollo de este proyecto han participado un alumno y el profesor. Las herramientas de comunicación que se han utilizado son el intercambio de información mediante correo electrónico y las reuniones con el profesor.

De las metodologías que se han planteado hasta ahora no hay ninguna que se ajuste bien a las necesidades de este proyecto por lo que se ha optado por tomar lo que más conviene de cada una de ellas. Las etapas han sido las siguientes:

A partir de una idea se ha producido un documento de visión que ofrece una perspectiva general de lo que será el proyecto y se ha expuesto al director del proyecto.

Mediante el análisis de alternativas hemos seleccionado la arquitectura que mejor permite integrar los dos sistemas. Se han evaluado distintas tecnologías existentes en el mercado que permiten la integración y se han descartado algunas.

Posteriormente, se ha realizado la captura de los requerimientos. Estos han sido redactados como sentencias que expresan un objetivo o una restricción que el sistema debe satisfacer, y deben ser comprensibles y no ser ambiguos.

En la siguiente fase se ha realizado la especificación del sistema. Se han utilizado artefactos de ingeniería del software como la notación UML. Cada uno de estos modelos muestra el sistema desde una perspectiva distinta, lo que ha aportado información que se ha utilizado en la implementación. La especificación se ha realizado mediante los siguientes modelos:

- Modelo de casos de uso.
- Modelo conceptual.
- Modelo de comportamiento del sistema.
- Modelo de los estados.

Tras la especificación, ya se dispone de la información suficiente para empezar con el diseño del sistema. Aunque en este punto ya sabemos que arquitectura vamos a utilizar ha sido de gran utilidad una herramienta la cual nos ha permitido generar un tipo de diagrama para ver la navegación entre las distintas pantallas que van a formar la aplicación web. En este punto se ha realizado una hoja de cálculo para la implementación del software adaptada a mis necesidades. En esta hoja de cálculo se han detallado los entregables como unidades que corresponden a una funcionalidad del sistema. Cada entregable está formado por subtareas el conjunto de las cuáles describe el procedimiento a seguir para conseguir implementar una funcionalidad en el sistema.

Un último comentario antes de finalizar este apartado es el siguiente: no se ha seguido un método de forma rigurosa y eso no quiere decir que se haya ido improvisando el diseño del software, sino que las distintas metodologías existentes no se han adaptado a este proyecto y se ha tomado lo que más podía interesar de cada una de ellas.

1.4 Organización de la memoria

Las etapas que componen esta memoria han sido las siguientes:

1) **Introducción** este capítulo sirve para ofrecer una visión de general de este proyecto e introduce algunos conceptos relevantes en este proyecto. Tras la visión general se expone los objetivos que debe cubrir el proyecto, aquello que se quiere lograr con todo el proceso.
2) **Análisis inicial** donde se exponen conceptos generales sobre SAP claves para el desarrollo de este proyecto.
3) En la **toma de requisitos**: se ha evaluado el sistema Web por una parte y SAP por otra. Las necesidades han sido las siguientes:
    a. La comprensión de los objetivos y necesidades del usuario en el aplicativo Web.
    b. El estudio de la información para la creación de un pedido y un cliente, así como, los procesos involucrados.

El **análisis funcional** del sistema ha consistido en la descripción del comportamiento externo del aplicativo Web desde el punto de vista del usuario y del entorno.

4) **Especificación:** llegados a esta etapa empezamos a utilizar artefactos en UML habituales de la especificación en la Ingeniería del software. Concretamente se presentan dos de ellos el modelo conceptual y el modelo por casos de usos.
5) Mediante el análisis de las **distintas alternativas** existentes en el mercado para realizar el aplicativo Web y su integración con SAP se han definido un conjunto de sistemas que podrían satisfacer las necesidades u objetivos y se han evaluado. De esta evaluación se ha determinado la arquitectura general del aplicativo Web que mejor satisface los requisitos en términos de Comunicación entre el sistema SAP y el aplicativo Web.
6) **Implementación** donde se documentan todos aquellos elementos que han sido desarrollados en el proyecto principalmente código y su correspondiente documentación.
7) **Pruebas** que se realizan al final del desarrollo. En él se presentan las distintas pruebas realizadas con los resultados esperados y los obtenidos.
8) Realización de un **Manual de usuario.**

1.4.1 Ámbito del PFC dentro de este proyecto.

Para desarrollar las etapas anteriormente nombradas ha sido necesario un doble trabajo. Por una parte investigar en SAP todos los procesos involucrados en el módulo de ventas, distribuciones y la gestión de los clientes. Por otro lado el desarrollo de un aplicativo Web que permita la integración con SAP y que cumpla con los objetivos anteriormente comentados.

2. Análisis inicial

2.1 Módulo basis.

El módulo BASIS es una parte fundamental dentro de SAP ya que provee al sistema de un entorno apropiado para que el resto de módulos instalados, FI (finanzas), MM (materiales), CO (controlling), etc. funcionen acordes a la base de datos y las aplicaciones de red y no supongan un riesgo para la seguridad de la información y los datos almacenados.

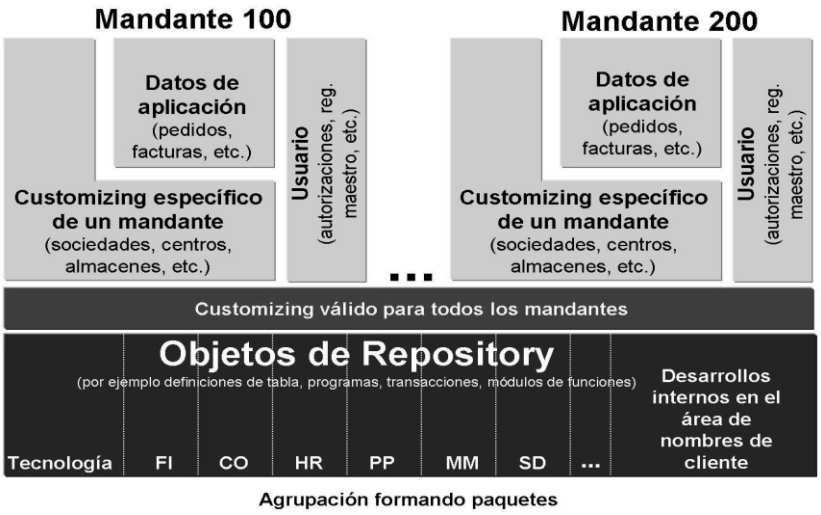
El módulo BASIS es el encargado de interactuar entre los servicios específicos del sistema y los servicios de las aplicaciones de negocio, administrando la seguridad y los accesos a los distintos módulos, permitiendo así, centralizar y controlar de forma eficiente los accesos otorgados a los usuarios, permisos y control de las transacciones realizadas.

Este capítulo introduce el módulo base como entorno de desarrollo central en sistemas SAP y algunas de sus herramientas que se han utilizado para el desarrollo de este proyecto.

2.1.1 Estructura de datos del sistema SAP.

El sistema SAP es un software empresarial que cuando se implementa hay que adaptarlo a las necesidades específicas de la empresa cliente. El proceso de adaptación del software se conoce como **Customizing**. Se utilizan parametrizaciones del Customizing para definir las estructuras organizativas del cliente en el sistema, tales como sociedades, canales de distribución y para fijar parámetros de transacciones SAP de tal forma que puedan reflejar procesos específicos de cliente.

Otro concepto específico de Sap es el **mandante**, el cual es una unidad independiente en lo que concierne a la empresa, la organización y los datos. Un mandante se caracteriza por tener su propio entorno de datos empresariales, así como sus propios datos variables, maestros y de usuario.



Estructura de datos de sistemas SAP

Para poder realizar este proyecto obviamente ha sido necesario contar con un sistema SAP que esté totalmente implementado y customizado. El sistema SAP el cual se utilizará para este proyecto es un sistema de pruebas con datos no reales y el cual está ya parametrizado.

2.1.2 El repositorio

El **Repositorio** es el almacén central para todos los objetos de desarrollo es válido en todos los mandantes. A continuación se definen en gran medida los objetos que forman parte del repositorio:

El **Diccionario ABAP** posibilita el acceso a todas las definiciones de datos utilizadas en el sistema SAP para describirlas y gestionarlas centralmente. Es un diccionario integrado y activo, en otras palabras, el Diccionario y ABAP se integra por completo en el entorno de desarrollo de SAP. La información del diccionario se registra una sola vez y se puede acceder a ella en cualquier momento y desde cualquier punto del sistema. El Diccionario ABAP proporciona automáticamente todo la información que se crea o modifica, asegurando así que los objetos en tiempo de ejecución son actuales y que los datos son consistentes y seguros. Mediante el diccionario creamos los siguientes objetos:

- . definiciones de objetos de base de datos (tablas, vistas).
- definiciones de tipos (estructuras, tipos de tabla)

- definiciones de servicios (Ayuda F1, Ayuda F4, objetos de bloqueo)

Workbench ABAP: es el entorno gráfico para escribir programas de aplicación y las aplicaciones Web.

2.1.2 Consecuencias y efectos de esta estructura de datos.

Para poder explicar las consecuencias de esta estructura de datos se deberá previamente explicar los entornos en que se desarrollan los procesos de negocio de una empresa:

¿Qué es un entorno productivo?

Es el ambiente donde la empresa opera, y sobre el cual se realizan todas las operaciones diarias de la organización, contiene toda la información sobre compras, pagos, cobros, contabilidad, y al mismo acceden los usuarios de las áreas específicas (Depto. de Contabilidad, Tesorería, RRHH, etc.).

¿Qué otros entornos necesitamos?

Cualquier empresa que tenga instalado SAP se encuentra constantemente realizando mejoras sobre su software, reparando errores, o implementando nuevas funcionalidades, además debe continuar con sus actividades de negocio .Dado que los objetos de repositorio no son específicos de un mandante, es necesario no utilizar el mismo sistema para el desarrollo y para la producción. El riesgo de una inconsistencia de datos es demasiado alto si se modifica cualquier objeto del repositorio en un entorno productivo. Por lo que se debe seguir una arquitectura de dos o tres sistema:

- Un **entorno de desarrollo** donde acceden los consultores de producto y desarrolladores que no poseen información del trabajo diario de la organización.
- Un **entorno de Calidad** al que acceden los consultores de producto, consultores funcionales y usuarios para probar el correcto funcionamiento del

programa o funcionalidad configurada en el ambiente de desarrollo pero sin alterar los datos del día a día de la organización.

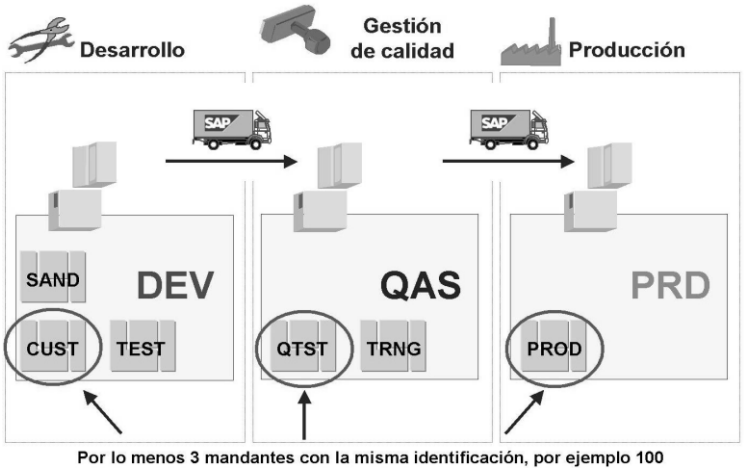
- Un **entorno de Producción** donde los consultores y desarrolladores no acceden, salvo en casos particulares y sólo como visualización, y es en este entorno donde la organización posee sus datos operativos.

El sistema de calidad debe contener en mayor o menor medida los mismos datos que el productivo. De esta forma se garantizara que el software desarrollado se comportara de igual forma en los distintos entornos.

Por este motivo gran parte de las empresas que tienen SAP instalado periódicamente copian los datos del sistema productivo a calidad. La alineación de los datos en distintos sistemas aumenta la fiabilidad de las pruebas y reduce el riesgo de inconsistencias cuando los objetos modificados entren en el sistema productivo.

2.1.3 Sistema de transportes.

En una arquitectura de dos o tres sistemas los transportes son necesarios para transferir de un sistema desarrollo a uno de calidad o productivo los objetos desarrollados.



La infraestructura de tres sistemas

Los objetos se transportan mediante la **orden de modificación**. El **Transport Organizer** es la herramienta de Sap que asigna un identificador único a la orden de modificación

de este modo se distinguen los distintos desarrollos que se están realizando en paralelo en un sistema Sap.

Una orden de modificación contiene todos los objetos relacionados lógicamente entre sí y se puede ejecutar únicamente de manera constructiva si permanecen juntos. Una orden permite por lo tanto el transporte y la gestión de desarrollos completos y finalizados constructivamente.

2.1.4 Asignación del equipo de desarrolladores a la orden:

Al comienzo de un proyecto de desarrollo el responsable del proyecto crea una orden de modificación. A continuación asigna a los desarrolladores implicados a la orden de modificación. El **Transport Organizer** crea automáticamente una subtarea para cada miembro del equipo de desarrollo asignado a la orden de modificación.

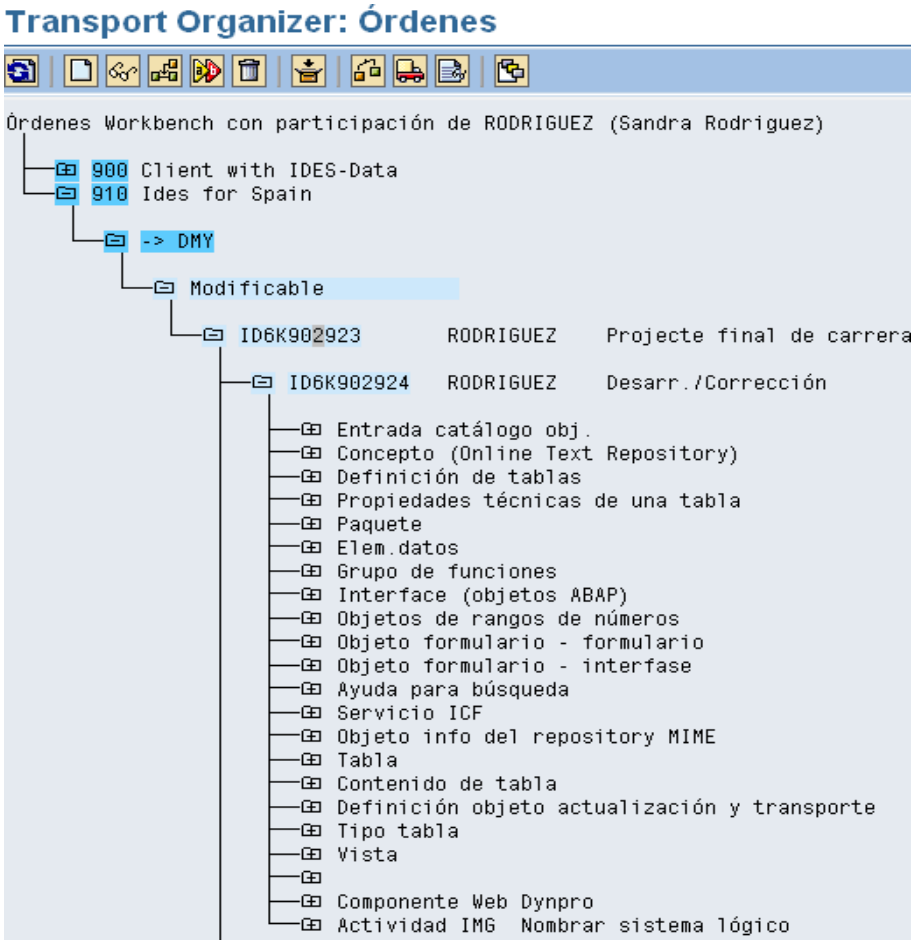
Siempre que un miembro del equipo asigna un objeto del Repositorio a la orden de modificación, dicho objeto queda registrado en su tarea.

Al final del proyecto la tarea contiene todos los objetos del Repositorio en los que ha trabajado dicho miembro del equipo. Cuando los miembros del equipo han acabado con su parte del proyecto de desarrollo, cada uno de ellos libera su tarea. De esta manera los objetos se transfieren de la tarea a la orden de modificación. Una vez que todos los miembros hayan liberado sus tareas, el responsable del proyecto puede liberar la orden de modificación.

2.1.5 Portabilidad

Una orden de modificación contiene todos los objetos del desarrollo creados modificados en el transcurso de un proyecto y se puede transportar de un sistema a otro. La orden de modificación contiene un atributo que indica cual es el sistema destino.

Al comienzo de la implementación del aplicativo Web se creó una orden de modificación y se asignó a la orden el equipo de trabajo, en este caso un único desarrollador. Los distintos objetos que forman el aplicativo Web se encuentran todos dentro de esta orden de forma que al transportar esta orden de un sistema SAP a otro se transfiere el aplicativo web.



Orden de transporte

En cuanto se libera una orden los objetos de Repositorio se copian de la base de datos fuente en un directorio a nivel de sistema operativo. A continuación se copian estos objetos en la base de datos del sistema de destino.

El software desarrollado se instala de forma fácil y sin coste alguno. Para instalar este aplicativo en cualquier sistema SAP se deberá definir previamente los datos del nuevo sistema en el sistema origen. En la orden de modificación cambiar el sistema destino para que apunte al nuevo sistema. Bastará con iniciar el proceso de transporte para que todos los objetos que forman el aplicativo se copien de la base de datos fuente a la base de datos del sistema SAP destino.

2.3 Módulo de ventas y distribuciones.

Las etapas del proceso que se describen en este capítulo muestran el modo de asignar y procesar pedidos de cliente en SAP. La gestión de los pedidos es un proceso

multidepartamental, por ello, la comunicación entre departamentos para mantener la información del pedido sin distorsiones requiere de niveles de información cada día mayores.

Uno de los principales objetivos en el aplicativo Web es que el cliente pueda generar pedidos en SAP a través de la interfaz Web. Por este motivo en este apartado vamos a introducir algunos conceptos y estructuras sobre la gestión de los pedidos en SAP, los objetivos que se van a tratar son los siguientes:

- Describir las estructuras de ventas y distribución.
- Describir las características de los pedidos y sus estados.

2.3.1 Estructura de la empresa en Ventas y Distribución.

La Compañía es la entidad legal e independiente con contabilidad propia. Es responsable de crear los balances y los estados de pérdidas o ganancias. La Organización de Ventas es la unidad logística que agrupa los requerimientos de ventas, siendo responsable de:

- Distribuir productos y/o servicios.
- Negociar las condiciones de venta.
- Representar a la empresa y a los productos que ella produce o comercializa.

El Canal de Distribución es el medio mediante el cual se les hace llegar los productos a los clientes, por ejemplo Internet, Exportación, Doméstico, etc. Representa a las estrategias definidas para la distribución y define las responsabilidades, sirve para diferenciar las estadísticas de venta, para manejo de precios, etc.

La Oficina de Ventas es la unidad usada para clasificar una zona o territorio con propósitos de ventas, de la cual dependen grupos de ventas y/o vendedores.

La Planta es la localidad donde los materiales son resguardados y usada igual en la manufactura.

- Desde la perspectiva de compras, la planta se usa para almacenar.
- Desde la perspectiva de Producción, la planta se usa para la manufactura.
- Desde la perspectiva de ventas, la planta sirve para distribuir materiales.

El Almacén es la sección asignada a una planta y el puesto de expedición es el elemento de mayor jerarquía dentro de los embarques.

2.3.2 Características de los pedidos.

En una compañía existen diversas transacciones para gestionar pedidos. En este proyecto únicamente nos vamos a centrar en la creación del pedido estándar.

El primer paso para poder realizar un pedido en SAP es tener un cliente de ventas, el cual debe tener informados:

- un solicitante que es quién genera el pedido de ventas.
- un pagador al cual se le remitirá la factura del pedido.
- un destinatario de mercancías que es quién recibe el pedido.

El pedido estándar es una solicitud que hace el solicitante a la empresa para la entrega de mercancías o servicios en un determinado momento. La solicitud se recibe en una organización de ventas que, a continuación, es responsable de cumplir con el pedido.

El pedido estándar es un acuerdo legal entre la organización de ventas y el solicitante para la entrega a precios claramente definidos, en cantidades exactas y en los tiempos acordados.

El escenario podría ser el siguiente: El cliente efectúa el pedido de mercancías por teléfono, correo postal, fax o correo electrónico, durante determinado tiempo y por determinado valor, la organización de ventas entrega la mercancía y factura al cliente por ello.

La cadena completa de de estados de un pedido (fase de preventas, pedido, entrega y facturación) conforma un historial y se conoce como el flujo de datos de un pedido.

Pero para poder gestionar el pedido este deberá contener información la cual se puede clasificar entre información específica, es decir, datos que serán distintos para cada pedido, e información general, datos que compartirán todos los pedidos, pero cuyo valor será distinto para cada caso.

2.3.2.2 Información específica

La información específica en SAP la conforman las líneas de pedido, es decir, los productos incluidos en el pedido. Una línea de pedido en SAP contiene muchísima

información pero únicamente se expondrá la más relevante para este proyecto que es la siguiente

- **Producto**: el artículo que se desea adquirir.
- **Cantidad:** el número de artículos.
- **Precio**: el importe del producto en el momento de ser añadido al pedido.
- **Fecha de entrega**: fecha estimada en que la línea de pedido será entregada al cliente.
- **Volumen**: el tamaño del producto

2.3.2.3 Información general

Por otro lado, SAP requiere guardar cierta información general cada vez que el cliente entra un pedido al sistema. A continuación se expone los datos más relevantes que conforman la información mínima de un pedido en SAP.

- **Identificador**: valor único para cada pedido para distinguir un pedido de otro.
- **Estado**: forma de saber en qué fase del proceso se encuentra un pedido.
- **Total:** el importe total del pedido a pagar por el cliente.
- **Fecha de entrada:** sirve para saber en qué momento el cliente confirma el pedido y éste se registra en el sistema.
- **Volumen total:** el volumen total del pedido.
- **Destinatario de mercancías:** el cliente que ha realizado el pedido.
- **Solicitante:** el cliente que recibirá el pedido.
- **Pagador:** a quién se le facturará el importe del pedido
- **Fecha de entrega:** la fecha en la que el destinatario de mercancías recibe el pedido.

2.3.3 Estados de un pedido.

- **Status de confirmación:** Indica que el pedido se ha creado en el sistema.
- **Status de entrega:** indica que el pedido ha sido entregado a su destinatario de mercancías.
- **Status de factura:** indica que el importe del pedido ya ha sido facturado a su pagador.

- **Status total de picking / almacenamiento:** indica que ya están listos todos los materiales que forman el pedido en la planta para ser cargados.
- **Status de bloqueo total:** mediante este estado se indica si el pedido está bloqueado y no se puede realizar ninguna operación con él.

3. Planificación

3.1 Planificación.

En proyectos grandes y complejos, donde durante algún momento de su desarrollo intervienen distintos recursos, es necesario llevar una gestión, control y asignación de recursos bien planificada. Este hecho se acentúa todavía más si las tareas que componen el desarrollo están interrelacionadas entre sí. Para llevar a cabo este tipo de planificación resultan de gran utilidad técnicas como los diagramas de Perth o Gantt. Sin embargo, en este caso se ha preferido usar otras herramientas similares que cumplen los mismos propósitos.

3.1.1 Planificación para el proyecto

El número de créditos es de 22,5. Si se cuentan unas 20h por crédito, el tiempo total es de 450h. La duración de este proyecto ha sido de dos cuadrimestres, ya que paralelamente a la realización del proyecto se ha estado trabajando en jornada completa. De modo que las fechas para la realización de algunas de las tareas son previsiones que a veces no se han podido cumplir.

Los métodos utilizados para realizar el proyecto han sido dos:

- unos informes semanales al director del proyecto para que conociera los avances y las dificultades. Este informe consistía en una hoja de cálculo que incluía las distintas tareas a las que se había dedicado el tiempo durante la semana y la cantidad de horas dedicadas a cada una. Acompañando a la hoja de cálculo debía ir un email explicando detalladamente lo que se había hecho y lo

que se pensaba hacer durante la semana siguiente. De este modo, la previsión concreta de horas se hacía de forma semanal, según las cuáles se mostraban las tareas no terminadas o la fecha estimada de finalización.

- Para el desarrollo del software, se ha diseñado una hoja de cálculo adaptada a mis necesidades. En esta hoja de cálculo estaban detallados los entregables que eran unidades que correspondían a una funcionalidad del sistema. Cada entregable estaba formado por subtareas el conjunto de las cuáles describía el procedimiento a seguir para conseguir implementar una funcionalidad del sistema. Cada entregable y subtarea tenía su fecha estimada de inicio y fecha estimada de finalización, así como su duración en horas.

Examinando atentamente mediante esta hoja hemos obtenido el cómputo global de horas, mediante la suma de la duración de todos los entregables. Pero dada la poca experiencia en gestión de proyectos y el no poder ajustarme a esas fechas debido a factores externos al proyecto, estas estimaciones han sido poco ajustadas a la realidad. Aún así, hacer la estimación ha permitido acotar tanto la duración probable del proyecto como el coste aproximado del mismo.

3.1.2 Etapas del proyecto

Al empezar el proyecto ya había una idea bastante clara de cuáles eran los objetivos a conseguir. Lo primero que se hizo fue un **análisis de alternativas** para elegir una arquitectura que nos permitiría la integración con SAP.

A continuación hubo una toma de contacto con el proyecto y con los **requisitos** preestablecidos mediante varias reuniones con el director del proyecto se estudiaron los requisitos globales del proyecto.

Una vez definidos los requisitos la siguiente etapa fue de **investigación**. Se estudiaron los procesos involucrados en SAP para generar clientes y pedidos, así como la información necesaria.

Posteriormente se paso a **especificar el sistema** se definieron el modelo conceptual, el de casos de uso y de comportamiento, se dedicó el tiempo necesario, ya que esa documentación serviría para los pasos siguientes.

La **implementación** consistió en seguir la hoja de cálculo que proporcionaba una guía para la construcción del software mediante el desarrollo de entregables.

El último paso ha sido la **documentación**, es decir, la creación de esta memoria. Algunas secciones ya se habían elaborado con anterioridad, como el análisis de los requisitos o el análisis de alternativas. También se incluye en este punto la preparación del material de presentación para la defensa del proyecto.

Tarea finalizada				
Tarea en realización				
Tarea sin empezar				
Entregables				
Entregable finalizado				
Partes				
Documentación				
Prototipo del sistema	5,0		26/07/2009	26/07/2009
	5,0	Diapositivas con la lógica de las pantallas	26/07/2009	26/07/2009
Memoria	123,0		29/07/2009	30/07/2009
	26,0	Análisis de alternativas	17/07/2009	17/07/2009
	20,0	Requisitos del sistema	16/07/2009	16/07/2009
	20,0	Especificación	19/07/2009	20/07/2009
	20,0	Diseño/Estudio de la herramienta	26/04/2010	01/05/2010
	15,0	Plan de Pruebas	01/05/2010	01/05/2010
	2,0	**Preparación material**		

Planificación 1

3.1.3 Planificación para el desarrollo del software.

Se ha realizado una planificación sobre la parte de desarrollo del sistema software. Esta segunda planificación se ha realizado para simular lo que sería una planificación real para un proyecto de este tipo, en el que no se tendrían en cuenta el resto de tareas propias de un proyecto de fin de carrera.

Como se ha comentado anteriormente las fases de esta planificación no reflejan exactamente cuando se realizaron realmente pero si a ha sido la metodología a seguir para el desarrollo del software.

El proyecto está formado por tres subsistemas independientes que han marcado la división en partes del desarrollo a realizar. Las partes son gestión de clientes, gestión de pedidos y autentificación. Estos subsistemas se dividen en un conjunto de

entregables los cuáles a su vez se dividen en pequeñas tareas. Una tarea puede estar finalizada, sin empezar o en realización. El conjunto de las tareas nos proporciona una "receta" la cual enumera los pasos a seguir para conseguir implementar una funcionalidad del sistema. Las fechas nos marcan el inicio y fin de cada tarea, la duración en horas es el esfuerzo a realizar.

Tarea finalizada				
Tarea en realización				
Tarea sin empezar				
Entregables				
Entregable finalizado				
Partes				
General	**Horas**		**Inicio**	**fin**
Diseño de la pantalla principal	10			
	5,0	Crear textos que informen de las partes del aplicativo	19/07/2009	19/07/2009
	5,0	Crear los plugs que te permitan navegar a las distintas pantallas ligados a los eventos desencadenados por los botones, links	19/07/2009	19/07/2009
Autentificación				
	5,0	Poner botón "valida" que ejecute la Consulta SQL a la tabla que almacena la relación usuario password	05/08/2009	06/08/2009
	8	**Pantalla**		
	1,0	Pruebas unitarias	06/08/2009	06/08/2009
Clientes				
solicitud de cliente	43,0		20/06/2009	26/08/2009
	8,0	Crear la pantalla para solicitud de alta con campos de entrada	20/06/2009	10/08/2009
	5,0	Comprobar que se han rellenado todos los campos obligatorios	11/08/2009	12/08/2009
	1,0	Comprobar que el NIF es correcto	13/08/2009	13/08/2009
	4,0	Crear el proceso "Avanzar" permite cambiar el estado a la solicitud	14/08/2009	15/08/2009
	15	Permitir anexar documentos relacionados con el cliente en la solicitud	16/08/2009	17/08/2009
	1,0	Comprobar si el cód. postal se corresponde con la población	18/08/2009	18/08/2009
	2,0	Crear desplegables para países, Poblaciones.	19/08/2009	20/08/2009
	1,0	Poner el botón "tramitar" para ejecutar el proceso "avanzar" y pasar al siguiente estado.	21/08/2009	22/08/2009
	5,0	Validar que la cuenta bancaria es correcta si procede.	23/08/2009	25/08/2009
	1,0	Pruebas unitarias	26/08/2009	26/08/2009
Tramitación	33,0		27/08/2009	08/09/2009
	17	Listado alv que mostrará todas las solicitudes de alta y en que paso del circuito están	27/08/2009	31/08/2009
	5,0	buscar de las tablas las solicitudes y en función del estado poner el icono correspondiente en el listado	01/09/2009	01/09/2009
	8,0	Poner botón a nivel de solicitud para visualizar la solicitud de alta para poder validar	02/09/2009	06/09/2009
	1,0	Plug de navegación enlazado al botón para navegar a la solicitud de alta y poder validar	07/09/2009	07/09/2009
	2,0	Pruebas unitarias	08/09/2009	08/09/2009
alta de cliente en SAP	38,0		09/09/2009	23/09/2009
	24,0	Realizar el proceso de alta cliente en Sap mediante " batch input"	09/09/2009	16/09/2009

	12,0	Gestionar los errores que pueda generar el alta de cliente en Sap mediante mensajes al workplace de empleados de ventas	15/05/2010	20/05/2010
	2,0	Pruebas	23/09/2009	23/09/2009
Gestión de pedidos	**Horas**		**inicio**	**Fin**
Crear pedidos	46,0		24/09/2009	15/10/2009
	16,0	Crear una listado de tipo “alv” que muestre el catálogo de productos de la empresa, permitir hacer búsquedas de productos dependiendo de distintos criterios de selección.	24/09/2009	28/09/2009
	3,0	volcar todos los productos de la base de datos al listado	29/09/2009	29/09/2009
	16,0	Crear el link "añadir producto" que permita añadir el producto al pedido	07/10/2009	09/10/2009
	5,0	Crear los plugs de navegación enlazados a los links	12/10/2009	15/10/2009
	5,0	Crear el botón logout para desconectar	20/10/2009	23/10/2009
	1,0	Crear el botón atrás para navegar a la pantalla anterior	24/10/2009	24/10/2009
Consulta pedidos	33,0		12/11/2009	17/11/2009
	15	**1**. Listado “alv” jerarquizado por pedido con las columnas	01/01/2010	08/01/2010
	5,0	**2.** Crear un campo que permita filtrar pedidos según el solicitante.	09/01/2010	09/01/2010
	8,0	**3.** Ayuda de búsqueda enlazada al campo solicitante	10/01/2010	11/01/2010
	2,0	**4.** Consulta SQL a las tablas que almacenan los pedidos según el cód. de cliente en SAP y pedidos con status abierto y no bloqueado	12/01/2010	12/01/2010
	3,0	**5.** Pruebas unitarias	13/01/2010	13/01/2010
alta de pedido en Sap	55,0			
	35,0	**1**. Realizar el proceso de alta pedido en Sap mediante " batch input"	14/01/2010	21/01/2010
	20,0	**3**. Gestionar los errores que pueda generar el alta de pedido en Sap	20/05/2010	25/05/2010
	2,0	**4.** Pruebas unitarias	28/01/2010	28/01/2010

Planificación 2

3.2 Análisis económico.

Una vez se tiene el cómputo de horas dedicadas al proyecto, se puede llevar a cabo el estudio económico de éste. En esta apartado se tiene en cuenta el coste del software desarrollado y el beneficio económico que aporta la instalación de este aplicativo en una empresa que tenga SAP instalado o que vaya a instalarlo. No obstante, por ser un proyecto de final de carrera hay que tener en cuenta que se trata de una estimación. La estimación se ha realizado de forma significativa para que aporte una aproximación al coste que generaría el desarrollo de este software en una situación real.

3.2.1 Coste del personal

Respecto al desarrollo, son necesarios dos perfiles: el de analista de sistemas y el de programador. Además, en todo proyecto suele haber un perfil de jefe de proyecto,

quién dirige las operaciones del equipo y toma las decisiones oportunas en un proyecto. En la actualidad dichos perfiles cobran sueldos del orden de los que se muestran en la tabla 2, donde se suponen una media de 240 días laborables por año y un coste de seguridad social a cargo de la empresa de un 33%.

Rol	Sueldo anual bruto	Coste anual	Coste/día
Jefe de proyecto	47.000 €	62.510 €	260.45 €
Analista	36.000 €	47.880 €	199.25 €
Programador	30.000 €	39900 €	166.25 €

Tabla 2

El proyecto dura 450 horas las cuáles se reparten 20 horas en reuniones. Tomaremos ese valor como tiempo de dedicación del jefe de proyecto y del analista. Aunque también contaremos las horas de análisis del sistema y especificación como horas de trabajo del analista, que dedicaría aproximadamente unas 120 horas. Si restamos esas horas al tiempo total obtenemos las horas del programador: 310. Los costes totales de personal se observan en la **¡Error! No se encuentra el origen de la referencia.**.

Rol	Horas	Días	Coste/día	Total
Jefe de proyecto	20	6,6	260,45€	1349,7€
Analista	120	15	199,25€	2.988,75€
Programador	310	38,75	166,25€	6442,18€
Total				**10.780,63€**

Tabla 3

3.2.2 Coste de hardware

El hardware utilizado para realizar este proyecto debe ser un ordenador, sobremesa o portátil. En todo momento durante el desarrollo será necesaria una conexión a Internet. Aunque no es del todo cierto que todo el coste de la conexión y el ordenador debería imputarse a este proyecto ya que en una empresa se desarrollaran distintos proyectos que utilizaran estos recursos. Por no contar con esta información se ha imputado el coste total de estos recursos a este proyecto la estimación para estos componentes es entonces:

- El ordenador desde el cual se realizó el desarrollo del software: Dell Latitude D520 349 **€**
- Router cableado de conexión a internet 45 **€**

El coste total de hardware es: **349 € + 45 € = 394 €**

3.2.3 Coste del software

Para la realización del proyecto se ha utilizado herramientas gratuitas: Firefox, Visual studio paradigm para realizar algunos modelos de objetos de la especificación y VPN. Por lo que no tiene coste alguno.

3.2.4 Coste total

El coste total tiene en cuenta los costes de las distintas partes que hemos comentado anteriormente y queda definido en la siguiente tabla:

Tipo de coste	Cantidad
Coste del personal	**10.780,63 €**
Coste del hardware	**449 €**
Total	**11.229 €**

Tabla 4

3.3 Condiciones contractuales de uso de licencias SAP.

Uno de los principales beneficios que aporta este proyecto está relacionado con la disminución del costo de las licencias contractuales.

Cualquier empresa que quiere instalar SAP deberá previamente hacer un estudio sobre las distintas tipologías de usuario que van a hacer uso del sistema. Esta es la base para calcular el precio total de las licencias a adquirir. Existen distintos tipos de usuarios (desarrolladores, administradores, usuarios de comunicación, profesionales) pero el principal usuario por el que se pagara más va a ser el usuario profesional el cual utilizara el sistema ERP de forma ilimitada o limitada en un entorno productivo.

El Tipo de usuario determina las licencias que son productivas y en base a esto SAP chequea que este número no exceda a las que figuran en el contrato de licencias. En

general el coste de las licencias para un usuario (como se define en el Acuerdo de Licencia) dependen de muchos factores

- Tipo de Usuario (desarrolladores, los usuarios en un sistema de producción), el acceso de sólo lectura.
- Gama de productos del contrato: SAP Business Suite, SAP ERP, Netweaver etc.
- El uso de características opcionales en el ERP (por ejemplo, las extensiones de Empresa o Industry Solution)
- Dentro de un contrato puede tener varios "instalaciones" (uno por cada tipo de entorno), cada uno de los cuales puede contener varios "sistemas", pero sólo un sistema de producción.
- Con frecuencia SAP proporciona también el gestor de base de datos y dependiendo del tipo de gestor existen diferentes porcentajes.
- Los grupos de clientes (partners de SAP) tienen sus variaciones propias de contrato. Dependiendo del país en que el sistema funciona, puede ser una prima significativa de los precios (introducir hasta 100%), en parte debido a los costos adicionales de traducción y lenguas extranjeras y el personal.

Como las políticas de precios y licencias de SAP son relativamente complejas, este análisis está basado en una representación aproximada, a grosso modo el precio por licencia de un usuario profesional es aproximadamente de 3.500,00 €. Adicionalmente hay que sumar las tasas de mantenimiento y actualización con un porcentaje promedio de 22% anual sobre el precio de adquisición de las licencias.

3.3.1 Beneficio económico

Supongamos el siguiente escenario, una nueva contratación de SAP como ERP con 100 usuarios en productivo: de los cuales unos 50 se dedicaran únicamente a introducir pedidos de ventas en el sistema. El ahorro que se generaría si se adquiere el software se basa en los siguientes puntos:

- Ahorro por disminución del número de las licencias a contratar

- Ahorro por la disminución del porcentaje a pagar cada año por mantenimiento de licencias.
- Ahorro por disminución de horas de trabajo del personal de administración de sistemas en la creación y el mantenimiento de usuarios (asignando roles, y autorizaciones).

La siguiente tabla refleja los cálculos que muestran en qué grado la empresa se beneficia económicamente. El ahorro del personal está en base a las horas que se deberían dedicar para la creación de los 50 usuarios y el coste del personal por hora.

Tipo de ahorro		Cantidad
Software	licencias	50 * 3.500 € = 17.500 €
Software	Mantenimiento de licencias anual	17.500 * 22% = 3850 €
Personal	Creación usuarios	40 h * 250 € = 10.000 €
Personal	Mantenimiento de usuarios	20 h * 250 = 5000 €
Total		**31.500 €**

Tabla 5

La Conclusión que se deriva de estos cálculos es la siguiente:
En el caso de que una empresa tenga que decidir en comprar 50 licencias o instalar el aplicativo Web, manteniendo las mismas cifras citadas anteriormente, sería obvio que la decisión final resultaría en la instalación del aplicativo, ya que en el primer año tendría un ahorro de 15.500 euros y 8.850 euros en los años restantes:

- 17.500 € Coste de licencias + 10.000 € Personal = 27.500 €
- 27.500 - 12.000 € coste del software = 15.500 €

Las empresas que ya tengan instalado SAP como ERP tendrán un ahorro anual de:

- 5000 € + 3850 € = 8850 €

El beneficio que genera la adquisición de este software aumenta de forma proporcional a la disminución de las licencias que se contratan.

4. Análisis de requisitos

4.1. Características de los usuarios

El B2B mayoritariamente está limitado a usuarios específicos pre-calificados con un conocimiento de la transacción que se va a realizar y del producto que se va a adquirir, Este factor influye directamente en los requisitos del sistema que se va a desarrollar. El sistema va a interactuar con tres tipologías de usuarios los cuales se van a distinguir mediante un sistema de roles que marcará las acciones que puede realizar cada uno. A continuación se introducen las distintas tipologías, aunque más adelante se realizará un análisis completo.

Las tres tipologías que van a interactuar con el sistema son:

- **el usuario cliente** es un usuario que ya ha sido dado de alta en SAP y puede realizar pedidos
- **el usuario empleado** no es cliente pero trabaja en la empresa gestiona las altas de los nuevos clientes, puede entrar pedidos para ellos y también puede rellenar solicitudes. Dentro del concepto usuario empleado existen distintas categorías que restringen las acciones a realizar.
- **usuario** ha rellenado la solicitud de cliente y se ha registrado en el sistema con la finalidad de llegar a ser cliente y poder realizar pedidos.

4.2 Solicitud de alta (formulario)

En un entorno B2B la información que se le pedirá al cliente será más amplia ya que se va a establecer una relación de confianza mutua. Necesitaremos adaptarnos a la información necesaria en SAP para poder generar un cliente. La información sobre el cliente la podemos dividir en dos grandes grupos:

- Información propiamente sobre el cliente y que rellenará el usuario en la solicitud de alta.

- Información a rellenar por el departamento de ventas. Estos datos dependerán de los criterios que haya definido la empresa para gestionar el departamento de ventas por internet y se rellenaran en el momento de la implantación.

4.2.1 Datos a rellenar por el usuario

Los datos que deberá rellenar el usuario se han agrupado en los siguientes apartados:

- **Dirección:** La dirección de un cliente se compone de un
  - Nombre del cliente, Nombre de la calle, Código postal, Nombre de la población, la región y la clave de país con la que se definen especificaciones que se utilizarán en la verificación de entradas como, la longitud del código postal o la longitud del número de cuenta bancaria.
  - La clave de idioma indica el idioma en que serán visualizados los textos, es el idioma de comunicación con el cliente.
- **Datos de control**: El Número de Identificación Fiscal (**NIF**) es la forma de identificación tributaria utilizada para las personas físicas con documento nacional de identidad (DNI) o número de identificación de extranjero (NIE) asignados por el Ministerio del Interior y las personas jurídicas.
- **Transacciones para el pago**: Una cuenta bancaria formada por:
  - **País** Identifica el país en el cual tiene su sede el banco. La clave de país determina las normas de verificación para los demás datos bancarios (p. ej., código bancario y número de cuenta bancaria).
  - **Clave de banco** En este campo se indica la clave de banco con la que se han archivado los datos bancarios en el país correspondiente.
  - **Cuenta bancaria** Este campo contiene el número bajo el cual se lleva la cuenta en el banco.
  - **Clave de control** Este campo contiene una clave de verificación para la combinación del código bancario y el número de la cuenta bancaria.

4.2.2 Datos a rellenar por la empresa.

Para gestionar las Ventas por internet la empresa deberá crear previamente este nuevo canal de distribución en SAP. Además deberá rellenar información específica

sobre el área de ventas. Todos los clientes que se creen en SAP mediante el aplicativo Web estarán sujetos a estas restricciones. La información a rellenar por la empresa será la siguiente y dependerá de cómo la empresa quiera gestionar este nuevo canal de distribución:

- **Moneda**: Moneda del cliente de un área de ventas. En la organización de ventas indicada se liquida al cliente en esta moneda.
- **Esquema de cliente** (para determinar el esquema de cálculo): Determina el esquema de cálculo (determinación de precios) que el sistema debe aplicar automáticamente cuando se debe crear un documento de ventas para un cliente.
- **Condición de expedición:** Estrategia general de expedición con la que se entregan mercancías al cliente.
- **Incoterms:** Fórmulas usuales de contrato que corresponden a las reglas establecidas por la Cámara de Comercio Internacional (ICC). Los incoterms establecen reglas reconocidas a nivel internacional, a las cuales se deben someter tanto el vendedor como el comprador para poder concluir de manera exitosa las transacciones comerciales.
- **Clave de condiciones de pago**: Clave a través de la cual se definen las condiciones de pago en forma de tipos de descuento y plazos de pago.
- **Tipo de impuesto** (IVA, federal, etc.) Identifica la condición que el sistema utiliza en la determinación de precios para establecer automáticamente los impuestos específicos de cada país.

4.2.3 Digitalización de la información

Para validar que todos los datos introducidos son correctos el usuario deberá anexar documentación digital clave que le identifique. Por este motivo se debe crear un apartado que permita anexar y visualizar todo tipo de archivos

La Ley Orgánica de Protección de Datos de Carácter Personal, (LOPD), es una Ley Orgánica española que tiene por objeto garantizar y proteger, en lo que concierne al tratamiento de los datos personales, las libertades públicas y los derechos fundamentales de las personas físicas. Su objetivo principal es regular el tratamiento

de los datos y ficheros, de carácter personal, independientemente del soporte en el cual sean tratados, los derechos de los ciudadanos sobre ellos. Por este motivo será necesario informar al usuario cuando este introduciendo sus datos.

4.3 Workflow de alta de cliente.

Un requisito de seguridad es que no todo usuario que rellena la solicitud de cliente se va a convertir en usuario cliente de SAP, podría ser que se quisiera descartar esta solicitud por distintos factores (datos incorrectos, insolvencia etc.).
Por este motivo existe la necesidad de crear un circuito de tramitación de las solicitudes de alta de cliente. El objetivo de este circuito es ratificar que todos los datos que se han rellenado en la solicitud y la documentación anexada son correctos.

En la mayoría de empresas validar que un futuro cliente es apto no depende únicamente de un empleado. Por este motivo existe la necesidad de crear el circuito y distinguir a los usuarios empleados responsables de la validación de los datos mediante un sistema de roles. Este sistema de roles marcará que trámite del circuito puede validar el usuario empleado que tenga ese rol asociado.

4.3.1 Sistema de roles, circuito activo y ruta

Un circuito estará formado por trámites y cada trámite tendrá uno o más roles asociados de esta forma un usuario empleado depende del rol que tenga asignado podrá validar uno o más tramites. Si el usuario empleado acepta o rechaza la solicitud la ruta marcará cual es el siguiente trámite a realizar. De esta forma se podrá parametrizar el proceso no restringiendo a un circuito concreto aunque siempre habrá uno circuito activo en el sistema.

Descripción del procedimiento

El usuario accede a través de Internet a la aplicación y desde aquí rellena los datos de la solicitud e inicia y registra la solicitud en el sistema.

1) Los responsables asociados a los distintos trámites del circuito recibirán un e-mail cuando la solicitud se encuentre en el trámite correspondiente al rol que tienen asignado. Este e-mail le informará de qué tiene una solicitud pendiente por validar.

2) El responsable accederá al aplicativo y validará que los datos introducidos son correctos y de esta forma la solicitud avanzará al siguiente trámite. No se podrá pasar al siguiente trámite si previamente no se ha validado el trámite anterior. Tanto si el responsable acepta o rechaza la solicitud la ruta marcará cual es el siguiente trámite a realizar. Pero siempre, el último trámite del circuito es generar el cliente en Sap. Además se debe poder rechazar definitivamente una solicitud y el usuario puede consultar online el estado de su solicitud en el sistema.
3) Cuando la solicitud se convierte en cliente de SAP el usuario recibe un email informándolo de que ya puede realizar pedidos, aunque como se ha comentado en el paso 2 esta información está disponible en el aplicativo.

4.4 Autentificación

Como se ha podido observar la gran mayoría de partes del aplicativo al tener acceso a SAP van a necesitar una autentificación. A continuación se exponen las partes y subpartes en una tabla que muestra el acceso al sistema.

Parte	Subparte	usuario
Gestión de clientes	Tramitación de clientes	Autentificación
Gestión de pedidos	Visualización de clientes	Autentificación
Gestión de pedidos	Generar/consultar pedidos	Autentificación
Gestión de clientes	Generar solicitud de cliente	Libre

Tabla 6

4.4.1 Visualización de solicitudes.

Para cumplir con la ley LOPD un usuario que no es empleado cuando acceda al aplicativo, concretamente a la tramitación de clientes visualizará únicamente su solicitud y la evolución de esta. Podrá consultar en todo momento en que trámite se encuentra su solicitud en el aplicativo.

En el caso de un usuario empleado por usabilidad del aplicativo no visualizará todas las solicitudes de alta que hay pendientes en el sistema sino que la visualización de solicitudes se restringirá al rol que tenga asociado. De esta forma sólo visualizará las

solicitudes que estén en el o los trámites asociados a su rol y que estén pendientes de validar. Además se le facilitarán medios para que pueda hacer búsquedas y filtrar.

4.4.2 Procedimiento de entrega de claves para la autentificación.

Para poder autentificarse el usuario deberá tener unas claves de acceso que serán el usuario y el password. El procedimiento de entrega de claves será distinto en función de la tipología de usuario. A continuación se describen los distintos procedimientos:

1) Si el usuario es empleado el mismo escogerá una clave para usuario y otra para el password y las comunicará para que as introduzcan en el sistema.
   a) Se comprobará que no hay otro usuario con las mismas claves.
2) El usuario cliente recibirá las claves en el momento del registro de la solicitud de cliente en el sistema.
   a) El aplicativo también enviará las claves por e-mail al usuario para que las pueda recordar.

Desde el sistema se debe poder gestionar las claves de modo que cualquier usuario pueda cambiar la contraseña y en el caso de que no la recuerde el aplicativo debe poder generar contraseñas y enviar el password al usuario.

4.5 Generar pedidos

La información que contiene un pedido se puede clasificar entre información específica, es decir, datos que serán distintos para cada pedido (datos de posición), e información general, datos que compartirán todos los pedidos, pero cuyo valor, obviamente, será distinto para cada caso (datos de cabecera).

4.5.1 Datos de cabecera de pedido

Para poder visualizar pedidos será necesario que el usuario se autentifique en el sistema. Mediante la autentificación se obtendrá el tipo de usuario y se distinguirá distintos procedimientos en función de si es usuario empleado o cliente.

- Si el **usuario es cliente** el procedimiento será el siguiente:

  Directamente se obtendrá el identificador que especifica de forma unívoca al usuario como cliente en SAP. (Es el código que SAP asignó al cliente cuando se

creó). Mediante este código se obtendrá de SAP toda la información necesaria relacionada con el cliente para generar el pedido.

- Si el **usuario es empleado** el proceso será el siguiente:

  El usuario empleado elige al cliente en nombre del cual va a realizar el pedido

  El aplicativo debe proporcionar mediante una ayuda de búsqueda todos los clientes que existen en SAP y que se registraron mediante el aplicativo. Esta ayuda de búsqueda deberá permitir al empleado encontrar de forma rápida al cliente. Los patrones de búsqueda que se utilizarán serán por código o por nombre de cliente pudiendo utilizar el carácter '*' para completar secuencias.

Los datos que se recuperan de SAP del cliente son los neceesarios para poder generar el pedido y se exponen a continuación:

- **Solicitante** es el cliente que ha pedido la mercancía o la prestación de servicios, en nuestro caso el solicitante siempre es el usuario cliente.
- **Destinatario de mercancías** es el interlocutor que recibe la entrega de las mercancías en nuestro caso siempre es el usuario cliente.
- **Pagador** es el interlocutor a quién se le va a realizar el cobro en nuestro caso es el usuario cliente.

Esta información identifica donde va dirigida la mercancía, quién realizó el pedido y a quién se le va a realizar el cobro, pero no es suficiente. Existe otra información no relacionada con el cliente pero necesaria para realizar cualquier pedido en SAP y que permite gestionar las ventas de una empresa.

4.5.1.1 Datos a rellenar por la empresa

Para gestionar las Ventas por internet la empresa deberá decidir esta información en el momento de implantación del aplicativo. Esta información se utilizará para crear todos los pedidos que se realicen desde el aplicativo y deberá poder ser consultada y actualizada. La empresa decidirá como quiera gestionar este nuevo canal de distribución de ventas por internet y rellenará esta información una única vez. La información a cumplimentar por la empresa será la siguiente:

- **Clase de pedido** Clasificación que distingue distintas clases de documentos de ventas, siempre serán pedidos estándar (no abonos, devoluciones, etc.)

- **Organización de ventas** es la unidad organizativa responsable de la comercialización de determinados productos o servicios.
- **Canal de distribución**: es el modo que permite la entrega de la mercancía o prestación de servicios.

4.5.2 Datos de posición de pedido

El usuario deberá escoger los productos que quiere añadir al pedido. El procedimiento se describe a continuación:

1) **Escoger el material**

   El aplicativo debe proporcionar mediante una ayuda de búsqueda todos los materiales que existen en SAP. Esta ayuda de búsqueda deberá permitir al empleado encontrar de forma rápida al material entre todos los que existen en SAP Los patrones de búsqueda que se utilizaran para poder encontrar el material será por código o por nombre de material, pudiendo utilizar el carácter '*' para completar secuencias de nombres.

2) Añadir producto al pedido.

   Cuando el usuario añada el producto al pedido se visualizará el precio unitario del material, el precio total por línea, así como total del pedido que se irá actualizando a medida que el usuario añade productos al pedido. Toda esta información sobre los precios del producto se deberá obtener de SAP.

3) Realizar pedido

   Cuando el pedido ya esté completo el usuario deberá poder tras generar el pedido en SAP desde el aplicativo.

4.6 Visualizar pedidos

El sistema ofrecerá la posibilidad a los usuarios clientes y empleados de visualizar pedidos. El procedimiento será distinto en función de si el usuario es empleado o cliente. A continuación se exponen los procedimientos.

- **Si el usuario es cliente** se obtendrá el identificador que especifica de forma unívoca al usuario como cliente en SAP (es el código que SAP asignó al cliente cuando se creó).

- **Si el usuario es empleado** previamente deberá escoger el cliente del cual quiere visualizar sus pedidos. El aplicativo debe proporcionar mediante una ayuda de búsqueda todos los clientes que existen en SAP. Esta ayuda de búsqueda deberá permitir al empleado encontrar de forma rápida al cliente entre todos los que existen en SAP. Los patrones de búsqueda que se utilizaran será por código o por nombre de cliente pudiendo utilizar el carácter '*' para completar secuencias.
    - Después de elegir el cliente el sistema obtendrá el identificador de cliente y mediante éste el sistema obtendrá de SAP todos los pedidos que tiene pendiente de entrega el cliente.

4.7. Requisitos funcionales

4.7.1 Solicitud de alta (formulario)

REQ_FORM_1: el sistema ofrece un formulario para obtener los datos identificativos del usuario que se quiere registrar.

REQ_FORM_2: la información a obtener del usuario es la mínima para poder generar mediante esta información un cliente en SAP.

REQ_FORM_3: el formulario debe ofrecer mecanismos para poder recuperar información de SAP y facilitar la introducción de la información.

REQ_FORM_4: el formulario debe ofrecer la posibilidad de poder adjuntar documentación que acredite la identidad del usuario.

REQ_FORM_5: el formulario debe ofrecer mecanismos para avisar al usuario en caso de error de datos.

REQ_FORM_6: el sistema debe comprobar que toda la información introducida por el usuario es coherente mediante los mismos procesos que realiza SAP.

REQ_FORM_7: el sistema debe proporcionar las claves al usuario en el momento del registro.

4.7.2 Workflow de alta de cliente

REQ_WORK_1: el sistema debe contar con un mecanismo que permita a distintos usuarios empleados validar que la información introducida por el usuario es correcta.

REQ_WORK_2: el sistema debe dar la posibilidad de generar un circuito que marque cuales son los trámites necesarios para crear un cliente.

REQ_WORK_2: el sistema deberá distinguir entre empleados que pertenecen a diferentes departamentos mediante un sistema de roles.

REQ_WORK_3: el rol marcará que trámites puede realizar el usuario empleado.

REQ_WORK_4: el sistema ofrecerá la posibilidad de consultar la información y documentación del formulario al usuario si su rol contiene el trámite en el cual se encuentra el formulario.

REQ_WORK_5: mediante la aceptación de los datos introducidos en el formulario el sistema debe avanzar.

REQ_WORK_6: mediante el rechazo de los datos introducidos en el formulario el sistema debe retroceder.

REQ_WORK_7: la ruta marcar cual es siguiente trámite en el caso de avanzar y cuál es el trámite previo en el caso de retroceder.

REQ_WORK_8: el sistema debe ofrecer un mecanismo para rechazar definitivamente una solicitud de alta en caso conveniente

REQ_WORK_9: el último paso del circuito es generar el cliente en SAP.

REQ_WORK_10: el sistema debe ofrecer un historial que permita mostrar la evolución de una solicitud de alta.

4.7.3 Autentificación

REQ_AUT_1: el sistema ofrecerá un procedimiento para que un usuario se autentifique (login).

REQ_AUT_2: el sistema ofrecerá un procedimiento para cambiar la clave de acceso.

REQ_AUT_3: el sistema ofrecerá un procedimiento para generar nuevas claves en caso necesario (olvido del usuario).

4.7.4 Generar pedidos

REQ_PED_1: el sistema ofrecerá una ayuda de búsqueda para que el usuario empleado encuentre fácilmente el cliente por el cual va a realizar el pedido.

REQ_PED_2: el sistema ofrecerá una ayuda de búsqueda para encontrar fácilmente el producto.

REQ_PED_3: los productos que puede comprar el cliente son los que hay en SAP.

REQ_PED_4: en el momento de introducir el producto en el pedido estará disponible el precio.

REQ_PED_5: el sistema debe dejar elegir la cantidad de producto.

REQ_PED_6: el sistema debe mostrar el total por línea y el total por pedido

REQ_PED_7: los totales se deben ir re calculando a medida que se van introduciendo más productos o se descartan.

4.7.5 Visualizar pedidos

REQ_PED_1: el sistema ofrecerá una ayuda de búsqueda para que el usuario empleado encuentre fácilmente el cliente del cual quiere visualizar sus pedidos.

REQ_PED_2: los pedidos que se visualizan son los que hay en SAP pendientes de entrega

REQ_PED_3: la información que se visualizará sobre el pedido se obtendrá de SAP y será el precio, la fecha de entrega, volumen y el estado del pedido.

4.8 Requisitos no funcionales.

4.8.1 Disponibilidad

REQ_DISP_1: la Web deberá estar siempre disponible (99% del tiempo)

4.8.2 Rendimiento

REQ_REND_1: se utilizará el servidor de balanceo de carga que proporciona el servidor de aplicaciones Web de SAP.

4.8.4 Seguridad

REQ_SEG_1: el sistema utilizará el esquema de autenticación de la propia web.

REQ_SEG_2: el acceso al formulario no requerirá login.

REQ_SEG_3: el sistema utilizará métodos de seguridad que proporciona el servidor de aplicaciones Web de SAP.

5. Especificación

5.1 Introducción

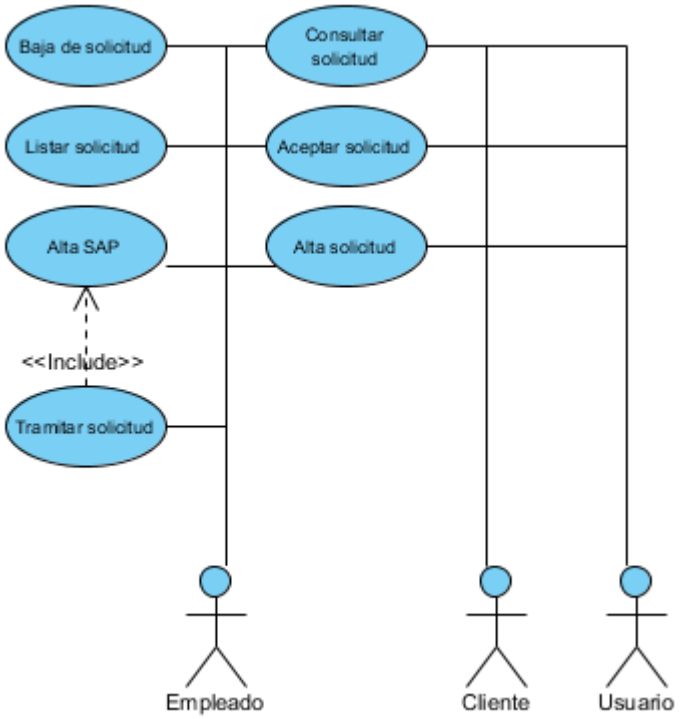
Una vez definido qué se quiere que haga el sistema (los requisitos), el proceso de diseño de software alcanza la etapa conocida como especificación.
Mientras que los requisitos trataban de reunir las funcionalidades y cualidades del sistema, la especificación trata de poner orden, por escrito y rigurosamente, de todo lo conseguido anteriormente. La fase de especificación define qué interacción tendrá el usuario con el sistema y de qué modo éste le responderá.

5.2 Modelo de casos de uso

El modelo de casos de uso recoge las secuencias de eventos que realiza un actor que usa el sistema para llevar a cabo algún proceso. Para ello se definen los actores, unidades externas al sistema que participan en la secuencia de eventos, y la propia secuencia de eventos que es el caso de uso. El sistema interactuará con tres actores el usuario, usuario cliente, usuario empleado.

5.2.1. Diagrama de casos de uso Gestión de usuarios

Esta agrupación hace referencia a la gestión de solicitudes en el sistema y al registro de la solicitud de cliente.



Caso de uso: Alta solicitud cliente

Actores: Usuario, Cliente, Empleado

Descripción: Rellenar un solicitud de cliente en el aplicativo Web.

Curso típico de eventos:

Acción de los actores	Respuesta del sistema
1.- El usuario inicia el proceso de alta **3**.- El usuario rellena la solicitud de cliente y envía datos	2.- El sistema le muestra el formulario de solicitud de cliente. 4.- El sistema comprueba los datos. 5.- El sistema guarda los datos y se crea el nuevo usuario del sistema. 6.- El sistema muestra las claves de acceso al usuario

Curso alternativo:

A4: Los datos no son correctos, se vuelve a paso 2.

Caso de uso: Baja solicitud

Actores: Empleado

Descripción: Borrar los datos de un usuario en el sistema Web.

Curso típico de eventos:

Acción de los actores	Respuesta del sistema
1.- El actor selecciona el usuario a dar de baja. 2.- El actor inicia el proceso de baja. 4.- El actor confirma.	3.- El sistema pide confirmación. 5.- El sistema elimina los datos del usuario.

Curso alternativo

A3: El actor no confirma y no se eliminan los datos.

Caso de uso: Listar solicitudes

Actores: Empleado

Descripción: Listar todas las solicitudes de cliente que hay en el sistema Web.

Curso típico de eventos:

Acción de los actores	Respuesta del sistema
1.- El actor selecciona ver la lista de solicitudes.	2.- El sistema muestra la lista de solicitudes.

Caso de uso: Consultar solicitud de un usuario

Actores: Empleado, Cliente, usuario

Descripción: Ver los datos de un usuario en el sistema.

Curso típico de eventos:

Acción de los actores	Respuesta del sistema
1.- El actor selecciona la solicitud.	2.- El sistema muestra la solicitud

Caso de uso: Tramitar solicitud

Actores: Empleado

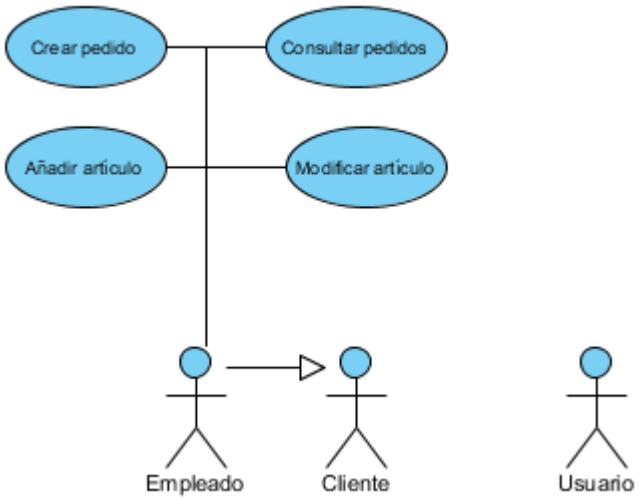
Descripción: Avanzar un trámite en el circuito creado con objetivo de que el usuario se convierte en cliente si se considera apto.

Curso típico de eventos:

Acción de los actores	Respuesta del sistema
1.- El actor selecciona la solicitud. 3.-El actor valida los datos 5.-El actor confirma	2.- El sistema muestra la solicitud 4.-El sistema pide confirmación

5.2.2. Diagrama de casos de uso Gestión de pedidos

En este grupo encontramos los casos de uso relacionados con los pedidos de productos realizados por o para los clientes.



Caso de uso: Crear pedido

Actores: Empleado, Cliente

Descripción: El pedido se traspasa a SAP

Curso típico de eventos:

Acción de los actores	Respuesta del sistema
1.- El actor le pide al sistema que quiere realizar el pedido 3.-El actor confirma	2.- El sistema le pide confirmación 4.- El pedido es realizado en el sistema SAP.

Curso alternativo:

A3: El actor no acepta y no se crea el pedido.

Caso de uso: Añadir artículo

Actores: Empleado, Cliente

Descripción: Un producto escogido por el usuario es añadido a la lista que conforma el pedido

Curso típico de eventos:

Acción de los actores	Respuesta del sistema
1.- El actor le pide al sistema que le muestre el catálogo de productos 3.- El actor elige un producto	2.- El sistema muestra los productos que hay en el sistema SAP

Curso alternativo:

A3: El actor no elige producto y se vuelve al paso 1.

Caso de uso: Consultar pedidos

Actores: Empleado, Cliente

Descripción: El actor consulta los pedidos pendientes de entregar

Curso típico de eventos:

Acción de los actores	Respuesta del sistema
1.- El actor inicia el proceso de consulta de pedidos	2.- El sistema identifica al actor y muestra los pedidos pendientes de entregar.

Caso de uso: Modificar carrito

Actores: Empleado, Cliente

Descripción: El actor elimina uno o más artículos del carrito

Curso típico de eventos:

Acción de los actores	Respuesta del sistema
1.- El actor inicia escoge un productos para eliminar 3. - El actor confirma	2.- El sistema le pide confirmación. 4.- El sistema elimina el producto del carrito

Curso alternativo:

A3: El actor no acepta y no se modifica el carrito.

5.3 Modelo conceptual

Tras la definición de los casos de uso, se puede proceder a modelizar el sistema.

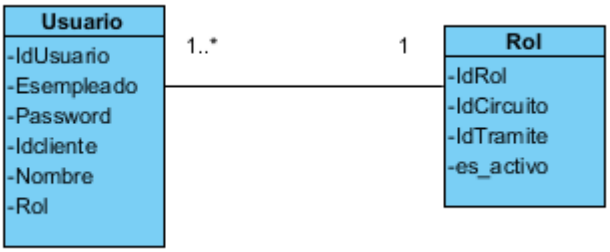
El modelo conceptual es esencialmente una representación de aquellos conceptos o entidades de la realidad que son trascendentes para el sistema. Permite además establecer la relación entre ellos y restricciones respecto a éstas.

5.3.1 Clases y atributos del modelo conceptual

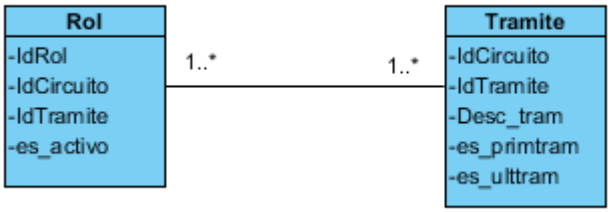
En primer lugar tenemos las clases y los atributos. Una clase es una entidad que representa un concepto de la realidad. Los atributos son las propiedades que tiene dicha entidad. Las clases contienen además operaciones, que, como su nombre indica, son aquellas acciones que se podrá realizar sobre la entidad y que provocarán un cambio en ésta. Una relación entre clases significa que existe algún tipo de vínculo entre ellas.

A continuación se definen las clases y sus atributos y se exponen las relaciones

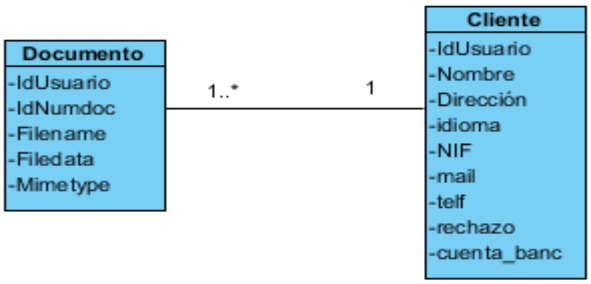
Asociación usuario y rol: No todos los usuarios serán clientes y por tanto la información de cliente del sistema se guarda por separado. El rol establece que trámites puede realizar un usuario.



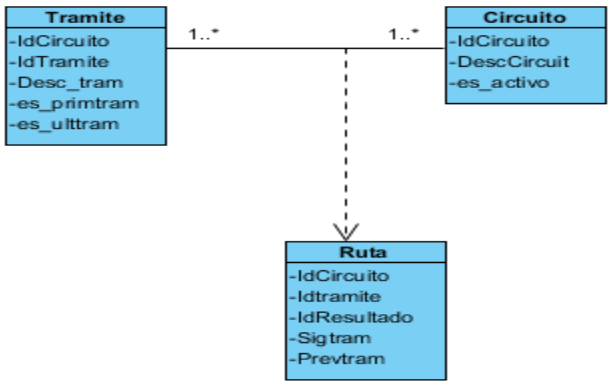
Asociación rol y tramite: un rol está formado por uno o más trámites.



Asociación documento y usuario: Los usuarios pueden adjuntar documentación en la solicitud de cliente

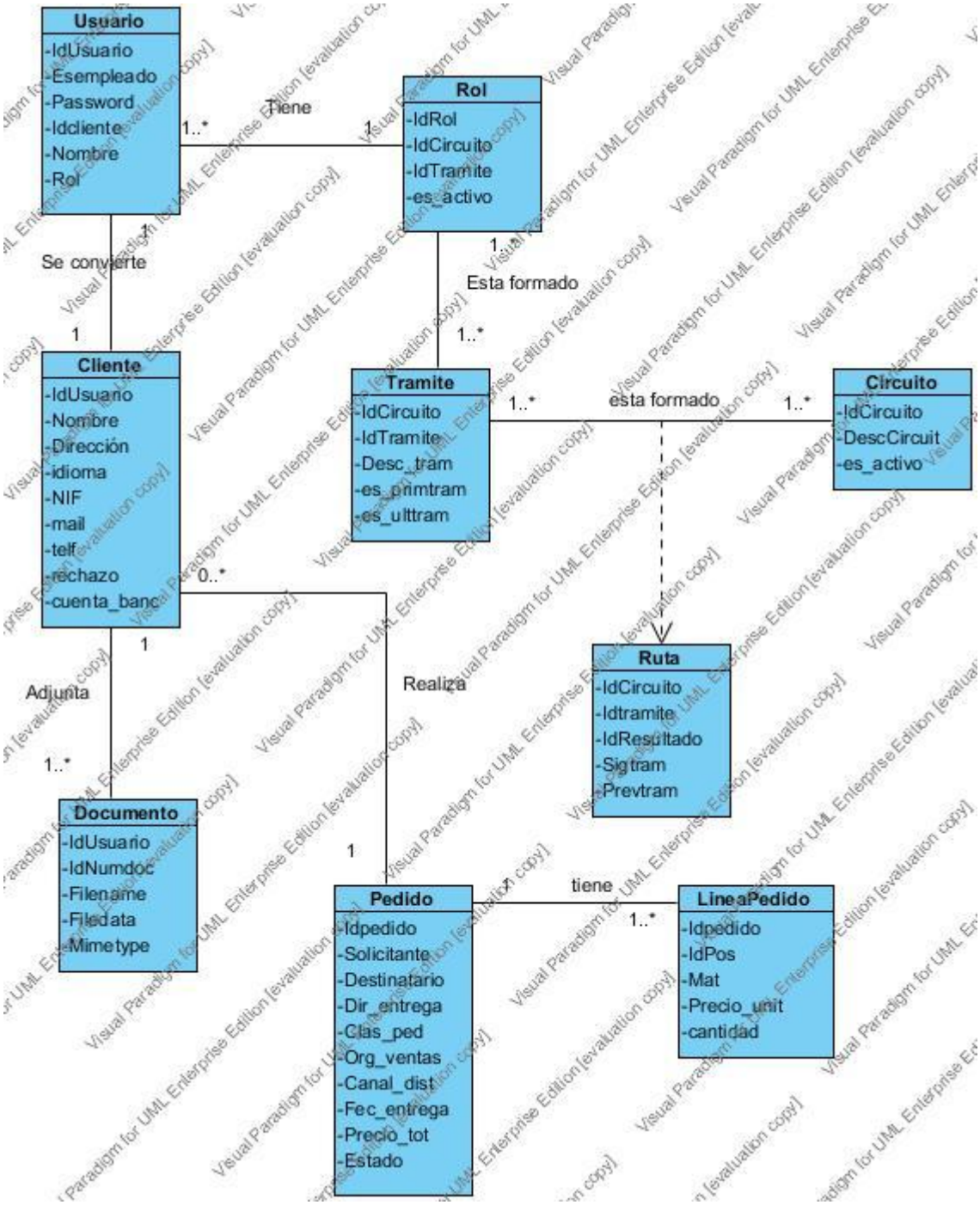


Asociación circuito, trámites y ruta: contiene información sobre qué circuito está activo en el sistema, los distintos trámites que lo forman y cuál es el primer y último trámite que se ejecuta. La ruta muestra la secuencia en que se deben ejecutar los distintos trámites dependiendo del resultado de la evaluación.



5.2.1 Diagrama de clases

A continuación se mostrarán todas las relaciones en el denominado diagrama de clases:



5.2.3 Restricciones de integridad.

El diagrama de clases no refleja totalmente la información que gestionará el dominio. Dichos datos son conocidos como restricciones y marcan las reglas que deben seguir las relaciones entre clases.

En nuestro diagrama son necesarias las siguientes reglas:

- Para todas las clases, el atributo id es clave primeria externa (no puede existir dos instancias de la misma clase con el mismo valor id).
- Sólo existirá un circuito activo en el sistema.
- El atributo es_primtram de la clase tramite marcará cual es el primer trámite que se ejecuta en el sistema.
- El atributo es_ultram de la clase trámite marcará cual es el último trámite que se ejecuta.
- El atributo es_activo de la clase Rol marcará si el rol se puede utilizar.

5.4 Modelo de comportamiento

El modelo de comportamiento, siguiendo en la línea del modelo de casos de uso, muestra como interactúa el usuario con el sistema. Además, permite, mediante el contrato de las operaciones, ver qué cálculos a nivel interno llevará a cabo el sistema en cada operación.

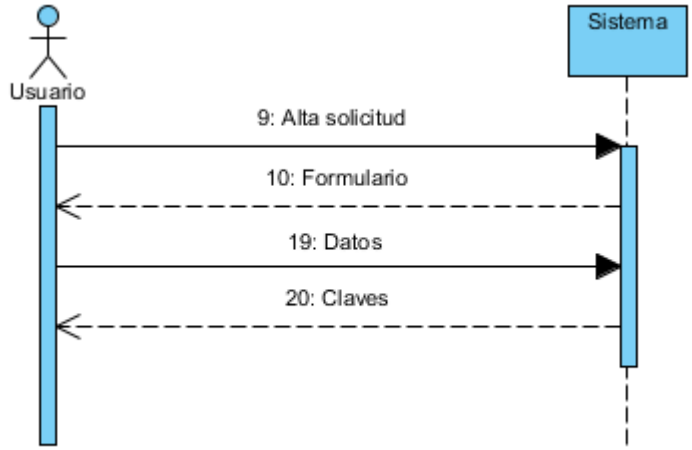
5.4.1 Diagramas de secuencia

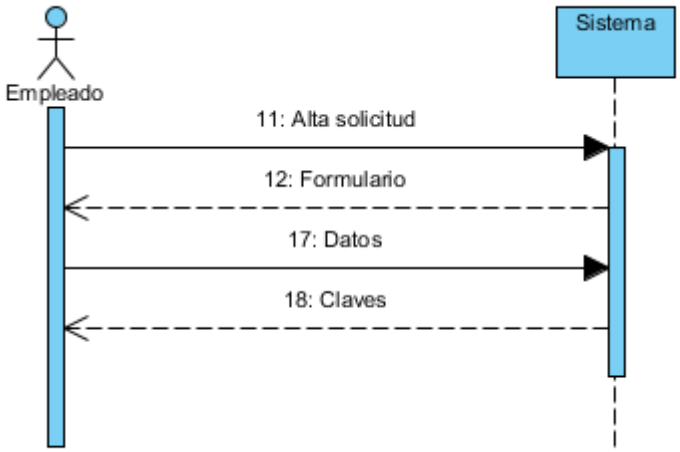
Los diagramas de secuencia ilustran la interacción entre los actores y el sistema, igual que ocurría con los casos de uso, pero centrados en un escenario particular. Cada curso de eventos relevante se refleja de un caso de uso en un diagrama, siendo éstos más precisos que los primeros.

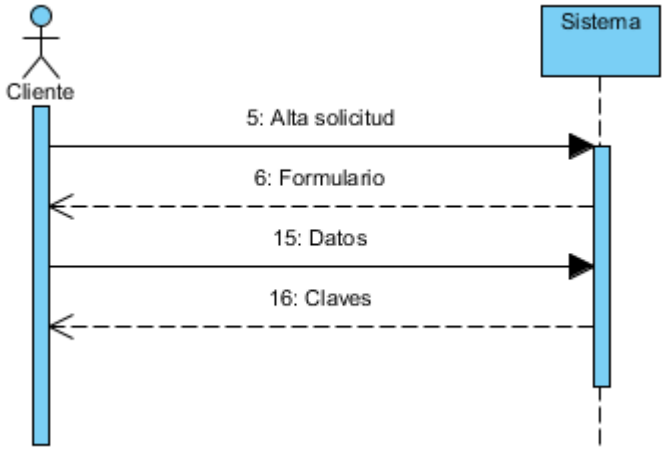
La estructura que siguen es la de un diálogo entre los actores y el sistema, que actúa como un actor más. Ambos se intercambian mensajes, que son las flechas que marcan el flujo de información.

5.4.1.1 Gestión de clientes

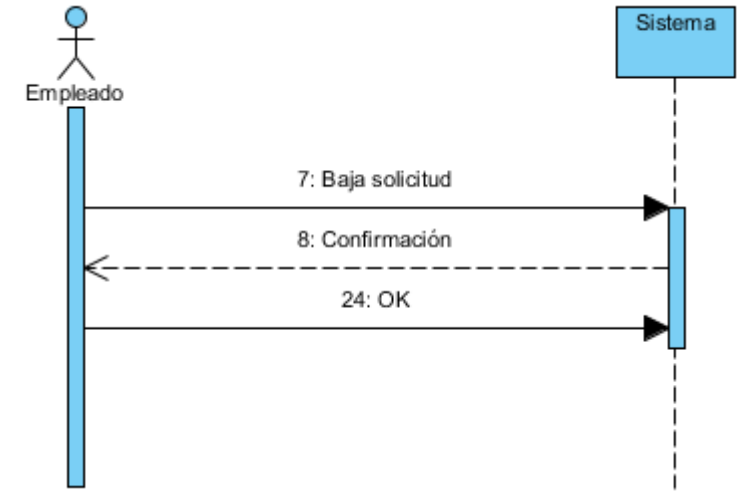
- **Alta de solicitud**



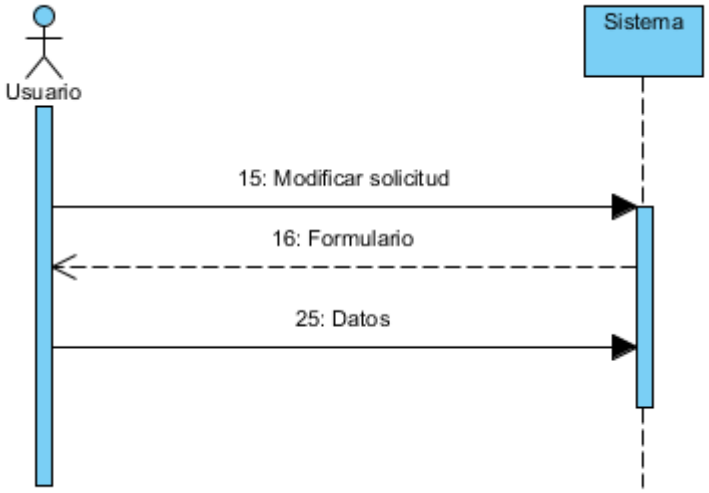


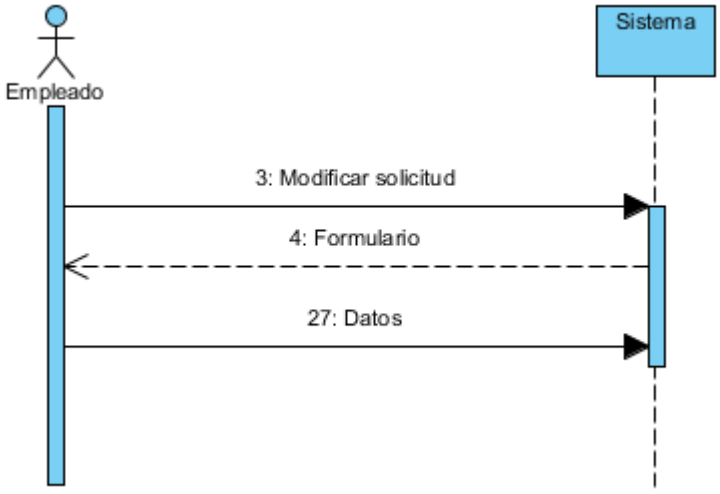


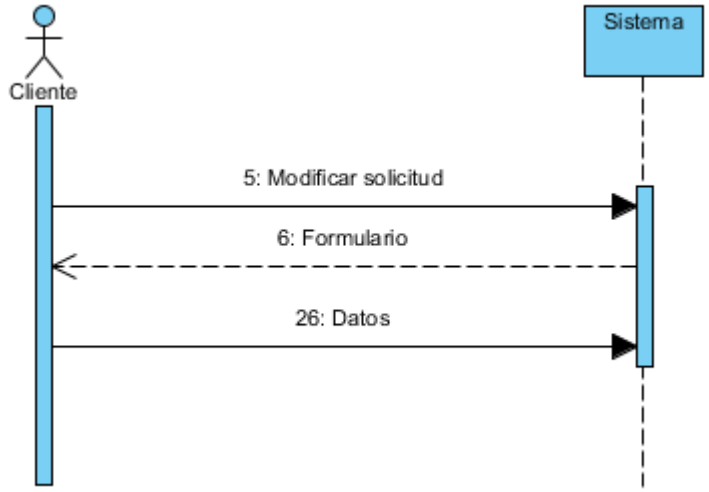
- **Baja de solicitud:**



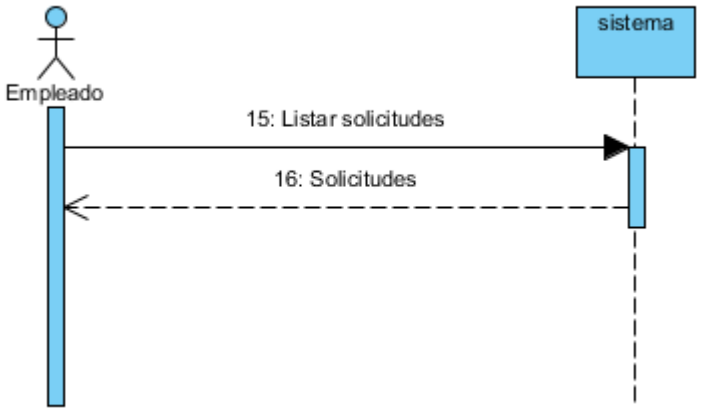
- **Modificar solicitud**



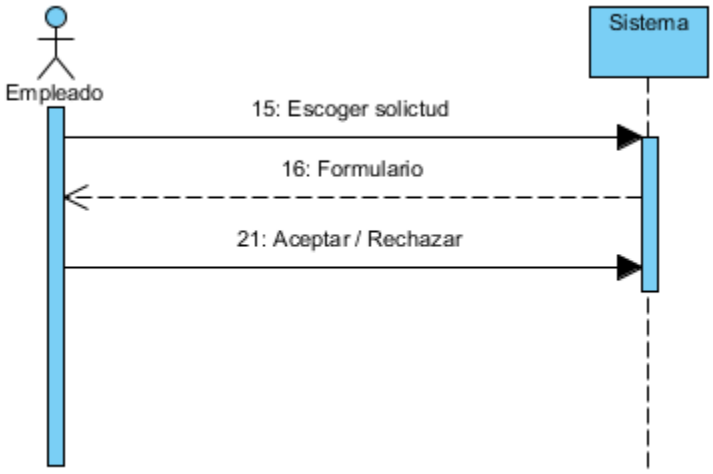




- **Listar solicitudes**

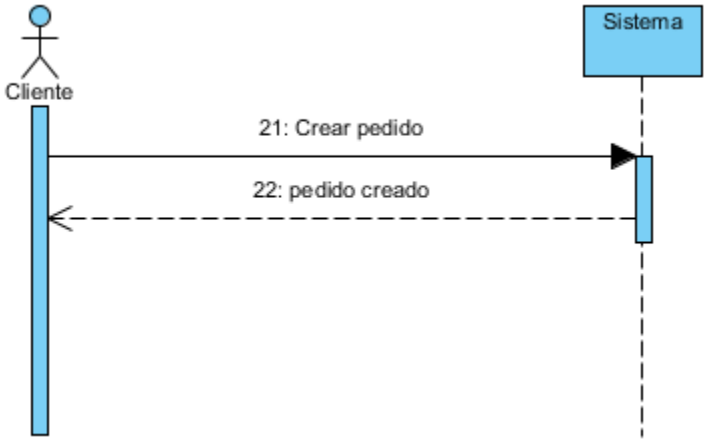


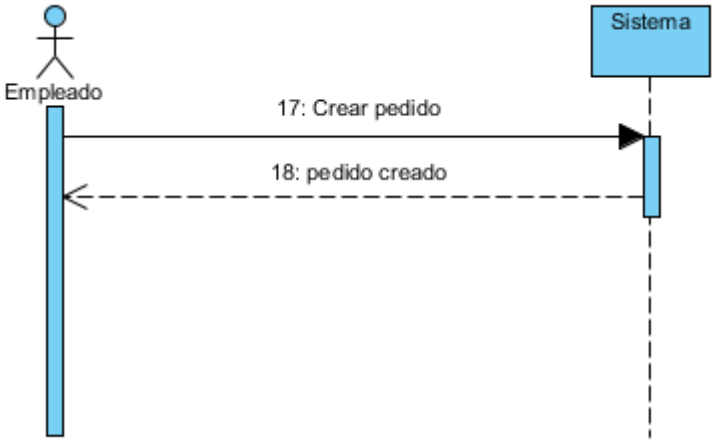
- **Tramitar solicitud**



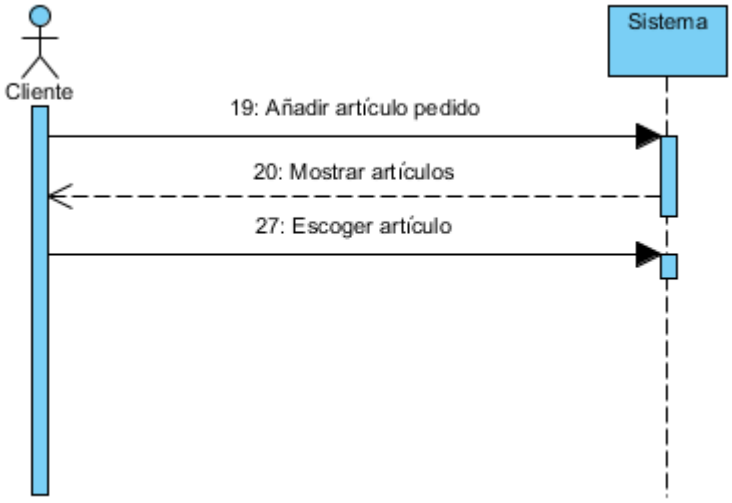
5.4.1.2 Gestión de pedidos

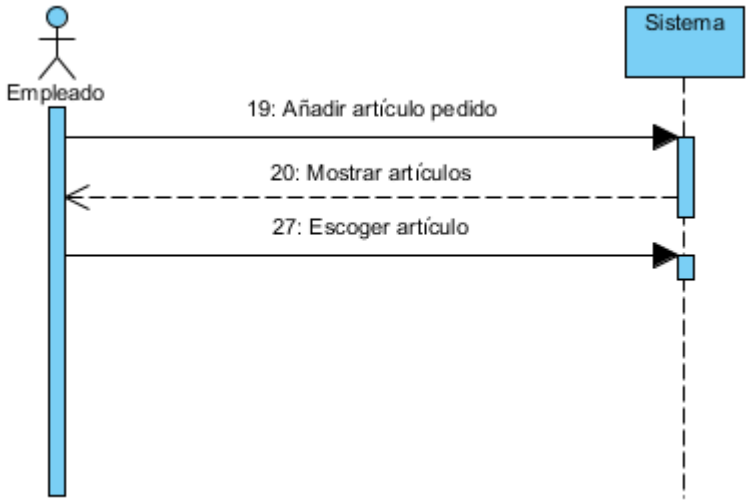
- **Crear pedido**



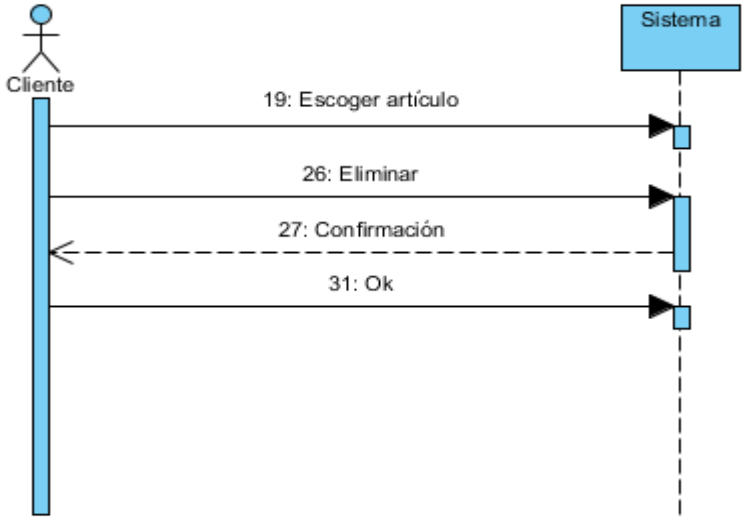


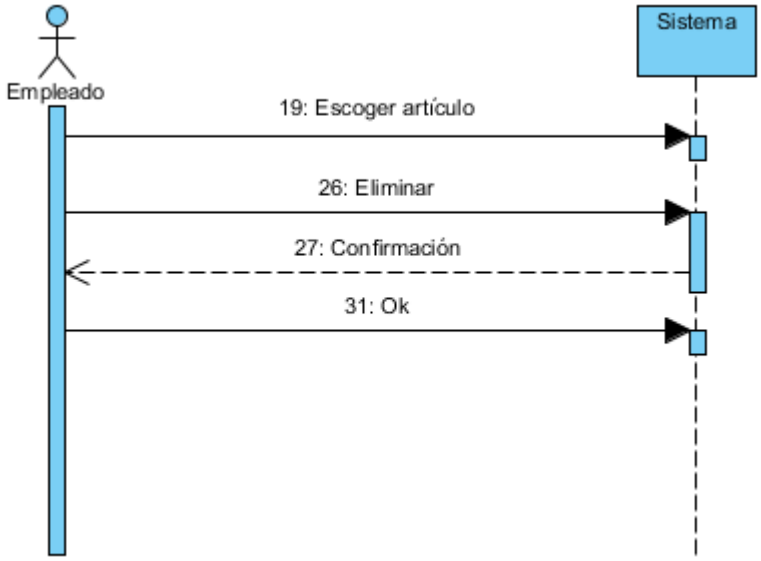
- **Añadir artículo pedido**





- **Eliminar artículo pedido**





5.5 Contrato de las operaciones

En los diagramas de casos de uso anteriores han aparecido multitud de operaciones. Sin embrago, no se ha detalla qué hace cada una.

El objetivo de este apartado es el de describir el comportamiento del sistema en cada operación, qué cambios de estado hay en la base de información y cuáles son las salidas que el sistema procesa.

La estructura que sigue el contrato de las operaciones es el siguiente:

• **Nombre:** cómo se llama la operación, seguido de, entre paréntesis, los datos de entrada y, tras dos puntos, los de salida.

• **Responsabilidades:** qué objetivo tiene la operación.

• **Excepciones:** caso de error.

• **Precondiciones:** qué condiciones han de cumplirse antes de la ejecución de la operación.

• **Pos condiciones:** qué condiciones se cumplen al terminar la ejecución.

• **Salida**: Resultado de la operación o objeto que crea y devuelve.

A continuación se listarán los contratos de las operaciones según el grupo al que pertenezcan.

5.5.1 Gestión de usuarios

Nombre: Alta solicitud (datos d): usuario

Responsabilidades: añadir una nueva solicitud al sistema.

Excepciones: si los datos no son correctos, indicar error.

Precondiciones: el usuario no existe en el sistema.

Pos condiciones: crea y guarda en el sistema una nueva solicitud de cliente.

Salida: u

Nombre: Baja de solicitud (usuario u):

Responsabilidades: eliminará una solicitud de cliente en el sistema.

Excepciones: si el usuario es cliente del sistema, indicar error.

Precondiciones: el usuario existe en el sistema y no es cliente todavía.

Pos condiciones: elimina la solicitud de cliente del sistema.

Salida:

Nombre: lista usuarios( ):

Responsabilidades: Listar los usuarios del sistema clientes y no clientes

Excepciones:

Precondiciones: El usuario es empleado

Pos condiciones: Crea una lista l con los usuarios del sistema

Salida: Lista

Nombre: Consultar información de usuario (usuario u): datos

Responsabilidades: Mostrar los datos de usuario.

Excepciones:

Precondiciones: El usuario es empleado.

Pos condiciones: Lee los datos d del usuario u.

Salida: datos.

Nombre: Tramitar solicitud (usuario u): datos

Responsabilidades: Avanzar un trámite en el circuito de alta de solicitud.

Excepciones: el usuario empleado no tiene el rol adecuado para avanzar.

Precondiciones: El usuario es empleado.

Pos condiciones: La solicitud de cliente avanza un trámite.

Salida:

5.5.2. Gestión de pedidos

Nombre: Añadir al carrito (producto p)

Responsabilidades: Agregar el producto p a la lista de productos que forman el carrito.

Excepciones: Si el producto p no existe, devuelve error.

Precondiciones: El producto p existe.

Pos condiciones: Añade p al carrito del cliente.

Salida: l

Nombre: Eliminar carrito (producto p)

Responsabilidades: Eliminar el producto p de la lista de productos que forman el carrito.

Excepciones: Si el producto p no existe, devuelve error.

Precondiciones: El producto p existe.

Pos condiciones: Elimina p del carrito del cliente.

Salida: l

Nombre: Crear pedido (carrito c)

Responsabilidades: Crear un pedido p a partir de los productos del carrito c.

Excepciones: Si el carrito c no existe o está vacío, devuelve error. Si el usuario que invoca la operación no es cliente, devuelve error.

Precondiciones: El carrito c existe y tiene productos.

Pos condiciones: Crea p con los productos de c.

Salida: p

Nombre: Confirmar pedido (pedido p, datos d)

Responsabilidades: Terminar el proceso de creación de un pedido añadiendo el resto de datos.

Excepciones: Si los datos d no son correctos, devuelve error.

Precondiciones: El pedido p existe.

Pos condiciones: Añade al pedido p los datos d.

Salida: p

Nombre: Consultar pedidos ( )

Responsabilidades: Mostrar todos los pedidos pendientes de ser entregados que tiene un cliente

Excepciones:

Precondiciones: El cliente existe.

Pos condiciones: Muestra la lista de pedidos de un cliente.

Salida: I

5.6 Modelo de estados

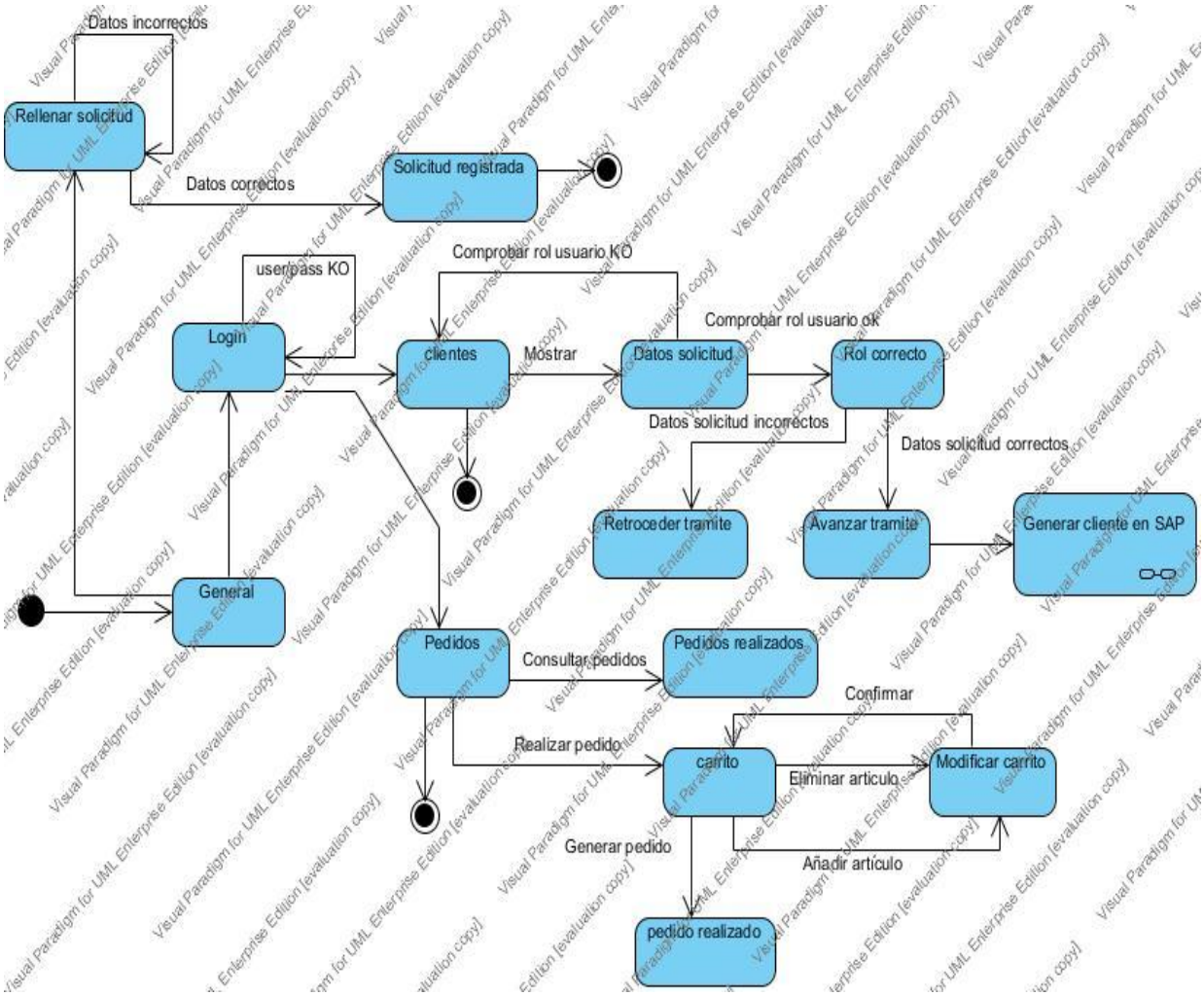
El modelo de estados tiene por objetivo mostrar los diferentes estados por los que pasa un objeto del sistema, usando para ello diagramas de estado. Los diagramas de estado constan de tres componentes:

• Evento: Aquello que requiere una respuesta del sistema.

• Estado: Condición de un objeto en el tiempo.

• Transición: Cambio de estado.

A continuación se muestra el modelo de estados para el sistema:



6. Diseño

6.1 Introducción

El diseño tiene como objetivo describir el "cómo" vamos a conseguir los objetivos.

Para ello se usan los llamados patrones de diseño. Un patrón de diseño es una abstracción de la solución, de modo que se puede aplicar a varios problemas. A menudo se pueden combinar uno o más patrones de diseño para obtener la solución al problema.

Sin embargo, hasta ahora no se ha hecho demasiado hincapié en el hecho de que se desarrollaba una aplicación integrada, pero lo cierto es que la fase de diseño se ve claramente condicionada.

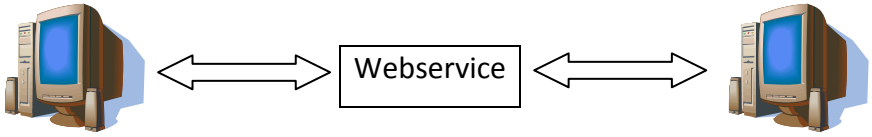
6.2 Análisis de alternativas

Existen varias alternativas al modelo propuesto en este proyecto. En esta etapa se han evaluado las distintas arquitecturas. A continuación se detallan y se expone el motivo por el cual se ha descartado.

6.2.1. Plataformas de comercio electrónico.

En una primera aproximación a la solución se pensó en intentar utilizar algún sistema ya existente en el mercado. ¿Porque desarrollar e implementar un sistema Web cuando en el mercado ya existen plataformas de comercio electrónico?

Sería más productivo invertir el esfuerzo en integrar SAP con la plataforma. Se pensó en un diseño basado en un servicio web que comunicara los sistemas. Se crearía un servicio web en SAP, su función sería la de obtener los pedidos existentes en la plataforma y pendientes de traspasar y posteriormente se crearían mediante los procesos que utiliza SAP.



Plataforma SAP

Los problemas que se derivan de esta solución se detallan a continuación

- **Replica de procesos**

  La pasarela de pago que proporciona una plataforma de e-commerce no se va a utilizar ya que los medios de pago son los que el cliente tiene definidos en SAP y el proceso de cobros a clientes ya se ejecutara en SAP.

  El aplicativo debe consumir los recursos que proporciona SAP para gestionar pedidos, sería redundante tener de nuevo una gestión de pedidos en la plataforma.

- **Replica de datos**

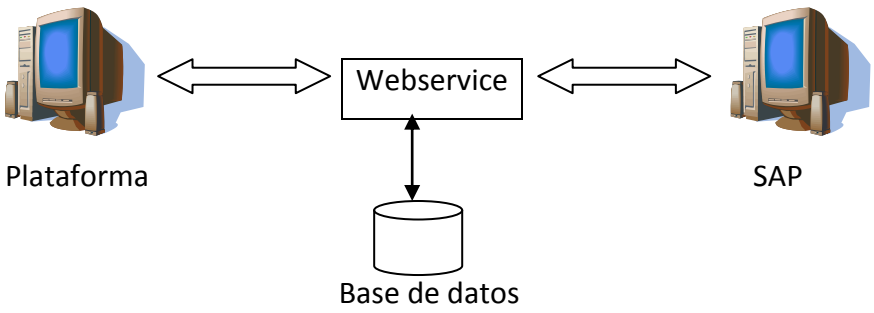
  Para instalar cualquier plataforma se requiere crear una base de datos que va a contener toda la información necesaria para la correcta gestión de los usuarios y pedidos de la tienda. Esta información ya existe en SAP por lo que estaríamos replicando datos además de tener dos bases de datos distintas con contenido similar.

6.2.1.1 Conclusión

El aplicativo pretende ser una forma de comunicación online entre SAP y el cliente y no replicar procesos que ya se ejecutan en SAP y tampoco replicar información.

6.3.2 Base de datos compartida

Para no tener dos bases de datos distintas con información replicada se barajo la opción de que los dos sistemas compartieran las tablas de base de datos involucradas en los procesos de negocio entre la plataforma e-commerce y SAP.



Esta solución se descartó rápidamente ya que cada uno tiene un modelo referencial de tablas distinto para una misma entidad.

Bajo esta arquitectura se debería investigar que tablas se actualizan y en qué orden para crear un pedido y un cliente en SAP.

En ambos casos, la falta de documentación por ambas partes de las tablas involucradas en los procesos es enorme. Esto aumentaba considerablemente el tiempo de ejecución del proyecto no pudiendo realizarse en el tiempo prefijado.

Bajo esta arquitectura también podían aparecer conflictos derivados por no cumplir factores de cualidad del software como son la Mantenibilidad e Integridad.

Cualquier cambio en el modelo de tablas por parte de SAP o Magento repercutiría en la solución creada teniéndola que adaptar de nuevo y con la posibilidad de dejar los datos inconsistentes.

6.3.2.1 Conclusión

Finalmente se descartó la integración de SAP con una plataforma y se optó por desarrollar un aplicativo Web. La elección del lenguaje Web está basada en la siguiente información:

La empresa a la que va dirigido este aplicativo ya tiene implantado SAP como ERP concretamente la versión SAP *NetWeaver.* Esta plataforma tecnológica convierte a SAP en un programa Web-enabled, lo que significa que estaría totalmente preparado para trabajar con él mediante la web.

Por este motivo se ha optado por utilizar una tecnología desarrollada por SAP para realizar Webs que permite fácilmente utilizar todas las funcionalidades disponibles en SAP en modo online.

6.3 Que es Web dynpro?

Esta herramienta es el modelo de programación estándar para construir interfaces Web de usuario en SAP NetWeaver.

Desde el punto de vista tecnológico, SAP Web Dynpro para Java y ABAP es un paso revolucionario en el desarrollo de interfaces de usuario basadas en Web. Es completamente diferente a cualquier paradigma de diseño utilizado previamente por SAP y representa un salto cualitativo en el desarrollo de la planificación de recursos empresariales basados en la Web. Las aplicaciones Web Dynpro se construyen utilizando técnicas de programación declarativa basado en el Modelo vista controlador

(MVC). Este modelo especifica cuáles son los elementos de interfaz de usuario y desde dónde van a tomar los datos. También define la navegación entres las distintas vistas. Todo el código para crear la interfaz de usuario es generado de forma automática dentro de un marco de tiempo de ejecución estándar. Por este motivo se ahorra tiempo en tareas repetitivas de codificación HTML.

Web Dynpro está diseñado para apoyar el desarrollo estructurado. El software se divide en unidades de modularización que son los componentes de WebDynpro, estos se pueden combinar para construir aplicaciones complejas.

6.3.1 Beneficios de Web Dynpro.

Un principio rector en la filosofía Web Dynpro es: cuantas menos líneas de codificación escritos a mano, mejor. Web Dynpro persigue este objetivo de la siguiente manera:

- Web Dynpro utiliza un modelo declarativo para la definición de un interfaz de usuario. A partir de esta definición abstracta, el entorno de desarrollo genera el código fuente necesario. El código a mano escrito todavía tiene su lugar, pero se limita a lo necesario para manipular los datos de negocio, no la interfaz. Además proporciona características técnicas tales como el apoyo a la internacionalización, y una separación limpia de la lógica de negocio y la interfaz de usuario. Esta separación se logra a través del modelo Vista controlador (MVC). Dado que las tareas repetitivas de codificación de elementos de interface han sido eliminadas, el desarrollador puede centrar su atención en el flujo de los datos.

6.3.2 Modelo vista controlador

Como hemos comentado anteriormente SAP Web dynpro está basado en el modelo vista controlador el cual está compuesto por tres partes fundamentales:

- **Modelo:** Corresponde a la representación de la información, en nuestro caso a las clases que se han mostrado anteriormente.
- **Vista:** Corresponde a la interfaz de usuario.

- **Controlador:** El controlador responde a eventos en la vista y ejecuta operaciones en el modelo.

El usuario interactúa con la vista cada vez que hace clic sobre un botón (por ejemplo, para enviar un formulario) genera un evento. Éste es captado por el controlador, quien invoca la operación asociada en el dominio y actualiza la vista o el modelo.

6.4 Arquitectura Web Dynpro

A parte de tener un modelo también es necesario tener una arquitectura en el sistema. Webdynpro separa la capa de presentación de la de los datos, esta división en capas es conocida como arquitectura en tres capas.

6.4.1 Visibilidad externa e interna en Web dynpro

La arquitectura de un componente Web Dynpro se puede dividir en dos partes: visibilidad externa e interna. La línea horizontal de trazos en la figura 1 separa las entidades que son visibles desde el exterior del componente de las que sólo son visibles desde dentro del componente. Las partes más visibles internas se pueden dividir en entidades visuales y entidades de programación. Las entidades visuales son las relacionadas con la interfaz de usuario y se comunicarán con el cliente.

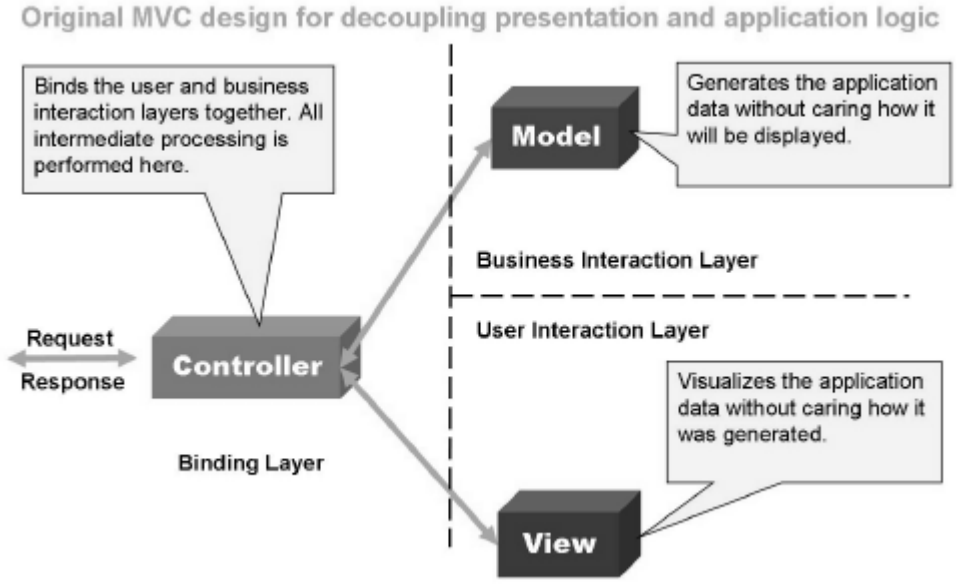


Figura 1

6.4.2 Capa de presentación

La capa de presentación se corresponde con la interfaz de usuario. El aspecto y diseño de la página dependen del trabajo realizado en esta capa. La capa se actualiza cada vez que el dominio le envía datos nuevos, lo que implica que su contenido sea dinámico. Como ya se ha comentado, existe la necesidad de separar la presentación del contenido en aplicaciones Web. Web dynpro va más allá y ofrece la posibilidad de usar lo que se denomina theme, un conjunto de plantillas o templates que dan aspecto y formato a todo el sitio Web. Los **themes** se instalan en el sistema como los módulos. Un **theme** incluye información como el juego de colores de la página, la distribución de sus menús, el tamaño y tipo de las fuentes, etc. convirtiéndose en una forma totalmente separada del código de funcionalidades de la Web.

6.4.1.1 Entidades de la capa de presentación

El diseño de esta capa está formado por vistas y ventanas, que representan una parte rectangular de una página que se le muestra al usuario. La vista contiene elementos de interfaz de usuario como campos de entrada y botones. La imagen completa que se envía al cliente puede ser creada por un único punto de vista, pero también puede ser una combinación de múltiples puntos de vista. Las combinaciones posibles de puntos de vista y el flujo entre los puntos de vista se definen en una ventana. Una ventana puede contener un número arbitrario de puntos de vista y una vista se puede incrustar en un número arbitrario de ventanas.

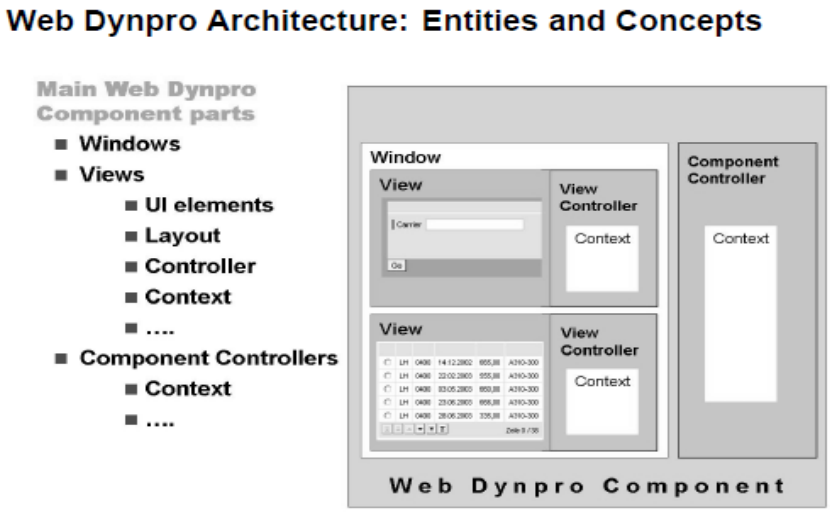


Figura 2

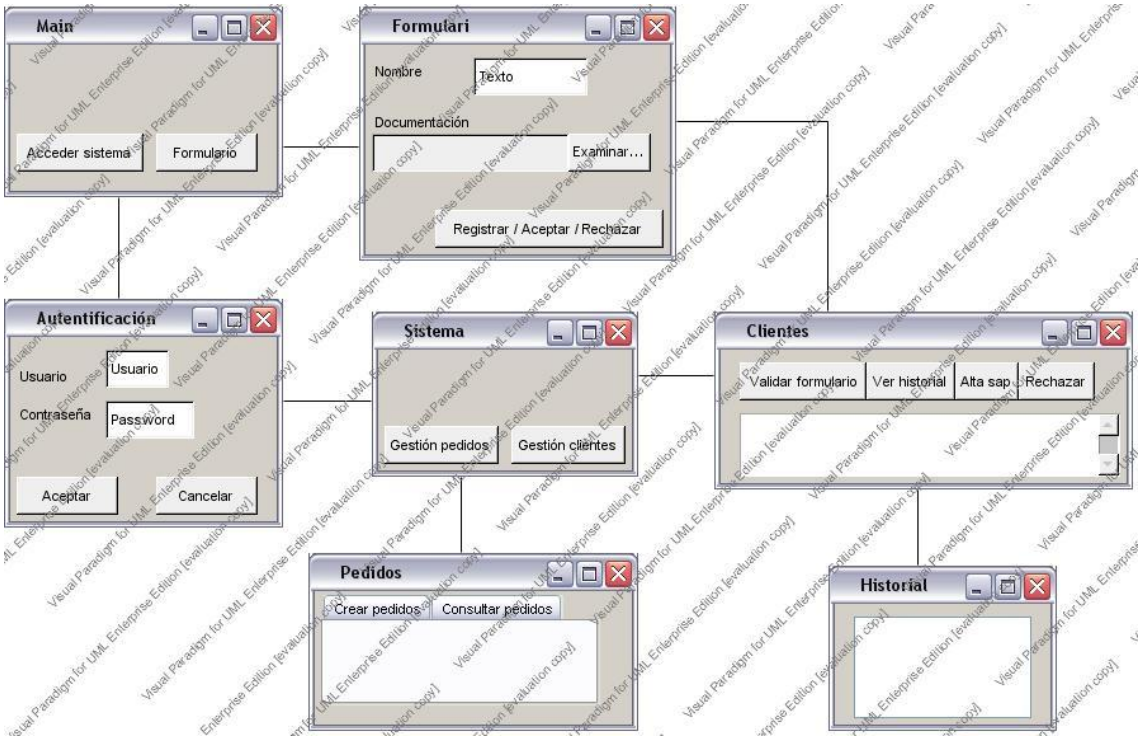
Una ventana define los puntos de vista que se muestran y de que forma la combinación de vistas puede ser cambiada por el disparo de un conector de salida. Por lo tanto, al crear una ventana, se definen dos cosas:

- Todos los puntos de vista posibles que pudieran existir en los componentes visuales de la interfaz que deben estar integrados en la ventana.
- Los enlaces de navegación entre los diferentes puntos de vista deben ser definidos y sólo una vista se muestra en el área de visión

6.4.1.2 Navegación entre vistas.

La navegación entre puntos de vista se desencadena por el disparo de los conectores de salida (por ej. un botón). Disparar un conector de salida provoca un suceso de navegación entre distintas vistas que se puede capturar mediante su correspondiente evento de navegación. Múltiples conectores de salida pueden ser disparados desde una vista. Pero para poder navegar de una vista a otra necesitamos tener un conector de entrada al punto de vista destino. Los conectores de entrada son métodos especiales del controlador de eventos que se suscriben a la navegación de eventos generados cuando se disparan conectores de salida. Los conectores salientes y entrantes se unen mediante enlaces de navegación. Los conectores de salida y los métodos de control de eventos relacionados con el conector de entrada pueden tener parámetros, lo que permite pasar los datos entre los puntos de vista.

El esquema de navegación entre vistas que se ha utilizado en este proyecto para la implementación se muestra a continuación. En este esquema las vistas no muestran todos los elementos de la interfaz, únicamente muestran aquellos que intervienen en la navegación. Los elementos de interfaz deberán estar asociados a los conectores de salida para poder navegar entre las vistas.



Esquema de navegación 1

6.4.2 Capa de dominio

La capa de dominio se corresponde con lo que antes hemos denominado “modelo”. Aquí se definen las clases mencionadas en el modelo conceptual (también conocido como diagrama de clases). La capa de dominio es la encargada de la gestión de las operaciones.

6.4.2.1 Contexto y transporte de datos.

El almacenamiento jerárquico de las clases se llama el contexto en la arquitectura Web dynpro y el valor de los elementos de la interfaz que permiten una entrada del usuario, tiene que estar conectado con el contexto. Este procedimiento se denomina enlace de datos. A través del enlace de datos, se establece un transporte automático de datos entre los elementos de la interfaz de usuario y el contexto. El transporte de datos entre elementos de la interfaz situados en diferentes puntos de vista se puede definir de una manera puramente declarativa.

El mapeo del contexto permite a un nodo de contexto en un controlador que se le suministre automáticamente los datos de un nodo de contexto correspondiente en otro controlador. Este es el principal mecanismo para el intercambio de datos entre las

distintas vistas.

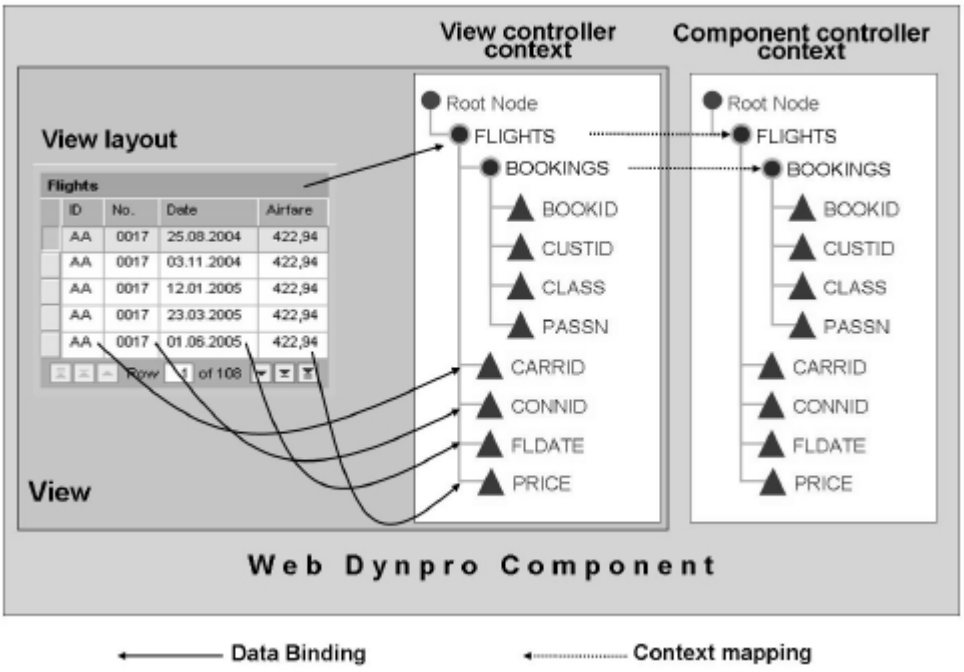


Figura 3

6.5 Diseño lógico

El diseño lógico establece la correspondencia entre el modelo conceptual y el esquema de tablas de la base de datos. En nuestro caso la correspondencia es directa por cada clase se ha creado una tabla cuyos campos son los atributos de la clase con sus correspondientes restricciones de clave.

6.6. Diseño físico

El diseño físico contempla el diseño a bajo nivel de las tablas de la base de datos. En SAP la definición de las tablas es independiente de la base de datos. La definición de tablas sirve de base para la creación de una tabla que tenga la misma estructura en la base de datos subyacente. El diccionario es la herramienta que proporciona SAP para convertir las definiciones a nivel de base de datos.

7. Implementación

En esta apartado exponemos los distintos componentes desarrollados, sus métodos, vistas, conectores de salida y entrada que se han creado en el aplicativo Web dynpro.

7.1 El controlador

El controlador es el componente que primero se ejecuta cuando se accede al aplicativo contiene los métodos públicos que se pueden invocar desde los otros controladores. Los nodos que se crean en este controlador son visibles para los otros controladores siempre que se establezca el mapeo del nodo.

7.1.1 Contexto

Los nodos creados en este controlador son la correspondencia con las clases que se han definido en el capítulo especificación, forman el contexto del Webdynpro o también llamado la capa de dominio según el modelo vista controlador. A continuación se muestra una tabla con todos los nodos creados. Además se han creado estructuras necesarias intermedias para almacenar información adicional.

Clase	Estructura
Cliente: los datos del cliente definidos en la especificación	Zwda_comprar: línea de pedido
Usuario: los datos del usuario definidos en la especificación	Manual: contiene el manual de aplicativo que el usuario se podrá descargar
Documento los datos de la documentación definidos en la especificación	Zstruc_alv_cliente: contiene los estados por los que pasa el cliente mediante iconos.
Historial los datos del historial definidos en la especificación	Zusuario_change: almacena usuario y contraseña en el momento de cambiar
Pedido: cabecera y líneas de pedido	Zusuario_forget: almacena usuario y contraseña en el momento que el cliente olvido su contraseña

Tabla 7: Clases definidas en el controlador

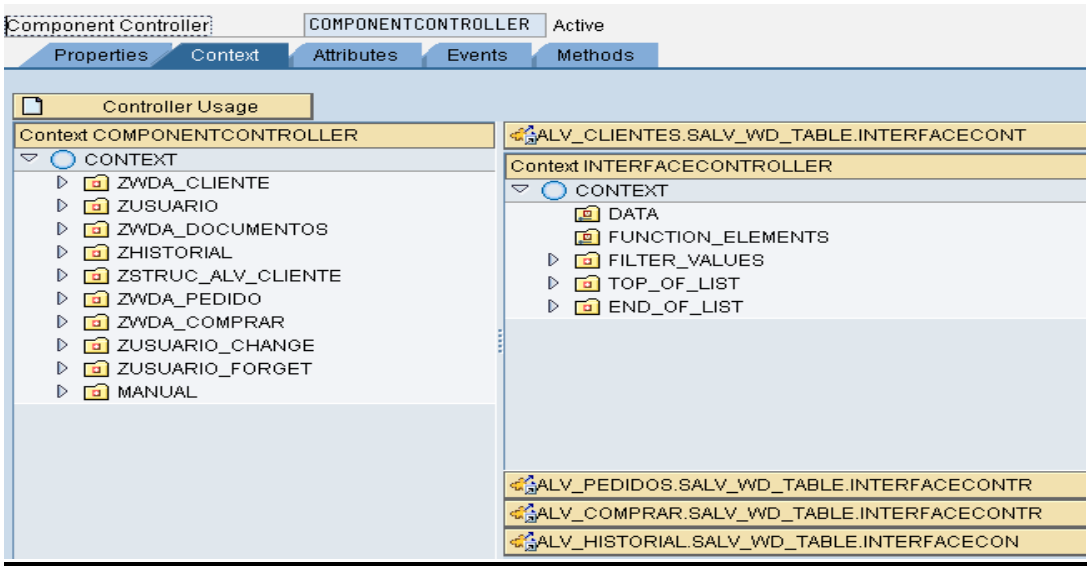


Figura 4: contexto del controlador

7.1.2 Métodos:

WDDOINIT

Este es el primer método que se ejecuta en el webdynpro mediante invocaciones a otros métodos se van creando las instancias de pedido, cliente e historial.

```
METHOD wddoinit.

  DATA: l_alv_pedidos, l_alv_ HISTORIAL  TYPE REF TO cl_salv_wd_config_table.
  DATA: l_ref_interfacecontroller TYPE REF TO iwci_salv_wd_table.
  DATA: l_alv_clientes TYPE REF TO cl_salv_wd_config_table.
  DATA: l_sigtramit TYPE char5, l_circuito TYPE char10,
        l_tramite_ini TYPE char5, l_tramite_fin TYPE char5.
  DATA lit_gestion TYPE ztab_zstructalv.
  DATA: l_alv_comprar TYPE REF TO cl_salv_wd_config_table.
  DATA lo_nd_zstruc_alv_cliente TYPE REF TO if_wd_context_node.

  wd_this->create_alv_HISTORIAL(
    CHANGING
      ref_interfacecontroller = l_ref_interfacecontroller
      alv_historial =  l_alv_historial                    ).

  wd_this->create_alv_clientes(
    CHANGING
      ref_interfacecontroller = l_ref_interfacecontroller
      alv_clientes =  l_alv_clientes                    ).
  wd_this->create_title_alv_clientes( alv_clientes =  l_alv_clientes  ).

  wd_this->create_functions_alv_clientes(
    CHANGING
    ref_interfacecontroller = l_ref_interfacecontroller
    alv_clientes =  l_alv_clientes                      ).

  wd_this->get_circuit_tramits(
    CHANGING
      circuit    = l_circuito                             " char5
      tramit_ini = l_tramite_ini                          " char5
```

```
      tramit_fin = l_tramite_fin                              " char5
      sigtramit  = l_sigtramit  ).

  wd_this->montar_ruta_circuito(
   tramite_fin  = l_tramite_fin
   alv_clientes = l_alv_clientes
   tramite_ini  = l_tramite_ini
   tramite_sig  = l_sigtramit
   circuito     = l_circuito    ).

   wd_this->rellenar_clientes_alv(
    CHANGING
      circuito   = l_circuito
      it_gestion = lit_gestion    ).

  lo_nd_zstruc_alv_cliente = wd_context->get_child_node( name = wd_this-
>wdctx_zstruc_alv_cliente ).

  CALL METHOD lo_nd_zstruc_alv_cliente->bind_table
     EXPORTING
       new_items            = lit_gestion.

  wd_this->create_alv_pedidos(
    CHANGING
      alv_pedidos =       l_alv_pedidos
       ref_interfacecontroller =     l_ref_interfacecontroller    ).

  wd_this->create_title_alv_pedidos( alv_pedidos = l_alv_pedidos  ).

   wd_this->create_functions_alv_pedidos(
    CHANGING
      ref_interfacecontroller =  l_ref_interfacecontroller
      alv_pedidos =  l_alv_pedidos ).

  wd_this->create_alv_comprar(
    CHANGING
      alv_comprar =    l_alv_comprar
      ref_interfacecontroller =    l_ref_interfacecontroller       ).

  wd_this->create_title_alv_comprar(
    alv_comprar =   l_alv_comprar                        ).

  wd_this->create_functions_alv_comprar(
    CHANGING
      alv_comprar =    l_alv_comprar
     ref_interfacecontroller =    l_ref_interfacecontroller ).

  wd_this->agregar_totales_alvcomprar( alv_comprar = l_alv_comprar ).

ENDMETHOD.
```

CREATE_ALV_CLIENTES

Este método crea el objeto listado de clientes el cual mostrará todos los clientes que hay pendientes de validar en el sistema.

```
METHOD create_alv_clientes.
  DATA  lo_cmp_usage TYPE REF TO if_wd_component_usage.
  lo_cmp_usage =   wd_this->wd_cpuse_alv_clientes( ).
  IF lo_cmp_usage->has_active_component( ) IS INITIAL.
    lo_cmp_usage->create_component( ).
  ENDIF.
  ref_interfacecontroller = wd_this->wd_cpifc_alv_clientes( ).
  alv_clientes = ref_interfacecontroller->get_model( ).
  alv_clientes->if_salv_wd_table_settings~set_visible_row_count( '10' ).
```

```
  alv_clientes->if_salv_wd_table_settings~set_fixed_table_layout( ).
  alv_clientes->if_salv_wd_std_functions~set_view_list_allowed( abap_false ).
ENDMETHOD.
```

CREATE_FUNCTIONS_ALV_CLIENTES:

Este método crea las funciones que se podrán realizar desde la gestión de clientes en el objeto listado:

- Ir a formulario para validar datos introducidos
- Alta en SAP del cliente
- Rechazar definitivamente cliente
- Visualizar historial

```
METHOD CREATE_FUNCTIONS_ALV_CLIENTES.

  DATA lr_functions TYPE REF TO if_salv_wd_function_settings.
  DATA: lr_function TYPE REF TO cl_salv_wd_function.
  DATA: lr_button   TYPE REF TO cl_salv_wd_fe_button.
  DATA: lr_image    TYPE REF TO cl_salv_wd_uie_image.
  lr_functions  = ref_interfacecontroller->get_model( ).
  lr_function = lr_functions->create_function( 'FORMULARI' ).
  CREATE OBJECT lr_button.
  lr_button->set_text( 'Solicitud cliente' ).
  CALL METHOD lr_button->set_image_source( 'ICON_ITS' ).
  lr_function->set_editor( lr_button ).
  lr_function = lr_functions->create_function( 'ALTA_SAP' ).
  lr_button->set_text( 'Alta cliente' ).
  CALL METHOD lr_button->set_image_source( 'ICON_IMPORT_TRANSPORT_REQUEST' ).
  lr_function->set_editor( lr_button ).
  lr_function = lr_functions->create_function( 'RECHAZO' ).
  lr_button->set_text( 'Rechazo definitivo' ).
  CALL METHOD lr_button->set_image_source( '~Icon/AlertMessage' ).
  lr_function->set_editor( lr_button ).
  lr_function = lr_functions->create_function( 'HISTORIAL' ).
  lr_button->set_text( 'Historial' ).
  CALL METHOD lr_button->set_image_source( '@0P@' ).
  lr_function->set_editor( lr_button ).
ENDMETHOD.
```

CREATE_TITLE_ALV_CLIENTES

Poner título en el objeto listado

```
METHOD CREATE_TITLE_ALV_CLIENTES.
  DATA: lr_header TYPE REF TO cl_salv_wd_header.
  DATA: lr_table_settings TYPE REF TO if_salv_wd_table_settings.
  lr_table_settings ?= alv_clientes.
  lr_header = lr_table_settings->get_header( ).
  lr_header->set_text( text-002).
ENDMETHOD.
```

FINALIZADO_CIRCUITO.

Mediante este método obtenemos si el usuario ya está dado de alta como cliente en SAP.

```
METHOD finalizado_circuito.
  DATA wa_historial TYPE zhistorial.
  SELECT SINGLE * FROM zhistorial
  INTO wa_historial
  WHERE identificador = identificador
  AND tramit          = 'ALSAP'
  AND circuit         =  circuito.
  IF sy-subrc = 0.
    finalizado = 'X'.
  ENDIF.
ENDMETHOD.
```

OBTENER_IDENTIFICADOR

Mediante este método obtenemos un identificador de usuario cuando este se registra en el sistema. La función NUMBER_GET_NEXT es una función que proporciona SAP y no es necesario implementarla.

```
METHOD obtener_identificador.
  CALL FUNCTION 'NUMBER_GET_NEXT'
    EXPORTING
      nr_range_nr             = '1'
      object                  = 'ZIDENTIDAD'
    IMPORTING
      number                  = identificador
    EXCEPTIONS
      interval_not_found      = 1
      number_range_not_intern = 2
      object_not_found        = 3
      quantity_is_0           = 4
      quantity_is_not_1       = 5
      interval_overflow       = 6
      buffer_overflow         = 7
      OTHERS                  = 8.
ENDMETHOD.
```

MONTAR_RUTA_CIRCUITO

Mediante este método montamos en el objeto listado la secuencia de trámites necesaria que deberá pasar cualquier solicitud de cliente.

```
METHOD montar_ruta_circuito.

  DATA l_image TYPE string, l_position TYPE i.
  DATA: lr_image  TYPE REF TO cl_salv_wd_uie_image.
  DATA l_name TYPE string, wa_tramites TYPE ztramites.
  DATA: lr_column_settings TYPE REF TO if_salv_wd_column_settings.
  DATA: lr_column TYPE REF TO cl_salv_wd_column.
  DATA: lr_header_alv TYPE REF TO cl_salv_wd_column_header.
  DATA: l_descrip TYPE string, l_language TYPE char2.
  DATA: it_tramites TYPE STANDARD TABLE OF ztramites,

  lr_column_settings ?= alv_clientes.
  l_position = 1.
  CALL FUNCTION 'CONVERSION_EXIT_ISOLA_OUTPUT'
    EXPORTING
```

```
      input            = sy-langu
    IMPORTING
      output           = l_language
    EXCEPTIONS
      unknown_language = 1
      OTHERS           = 2.
  IF sy-subrc <> 0.
    MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
            WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
  ENDIF.
  SELECT  *  INTO CORRESPONDING FIELDS OF TABLE it_tramites
  FROM ztramite WHERE circuito  =  circuito.
  SORT it_tramites BY pos.
  l_name =  tramite_ini.
  CALL METHOD lr_column_settings->get_column
    EXPORTING
      id    = l_name
    RECEIVING
      value = lr_column.
  LOOP AT it_tramites INTO wa_tramites.
    CREATE OBJECT lr_image.
    IF sy-tabix = 1.
       l_position = wa_tramites-pos + l_position.
    ENDIF.
* recuperem la descripcio del tramit
    SELECT SINGLE  descrip FROM ztramites_text
    INTO l_descrip WHERE circuito = circuito
    AND tramite  = wa_tramites-tramite
    AND idioma   = l_language.
    l_name = wa_tramites-tramite.
    lr_column = lr_column_settings->get_column( l_name ).
    lr_image->set_source_fieldname( l_name ).
    lr_column->set_cell_editor( lr_image ).
    lr_column->set_position( l_position ).
    lr_header_alv = lr_column->get_header( ).
    lr_header_alv->set_text( l_descrip ).
    l_position = l_position + 1.
  ENDLOOP.
* ultima columna
  l_descrip = 'Alta en Sap'.
  lr_column = lr_column_settings->get_column( 'ALSAP' ).
  lr_column->set_position( l_position ).
  lr_header_alv = lr_column->get_header( ).
  lr_header_alv->set_text( l_descrip ).
* PRIMERA columna
  l_descrip = 'Identificador'.
  lr_column = lr_column_settings->get_column( 'IDENTIFICADOR' ).
  lr_column->set_position( 1 ).
  lr_header_alv = lr_column->get_header( ).
  lr_header_alv->set_text( l_descrip ).
ENDMETHOD.
```

RELLENAR_CLIENTES_ALV

Cargamos todos los clientes con trámites pendientes de validar en el objeto listado de clientes.

```
METHOD rellenar_clientes_alv.

  DATA:  it_historial TYPE STANDARD TABLE OF zclihist,
         lit_bloqueos TYPE STANDARD TABLE OF zbloqueos,
         wa_bloqueos  TYPE zbloqueos, wa_historial TYPE zclihist,
         wa_historial_aux TYPE zhistorial, wa_gestion TYPE zstructalv.
  DATA lo_nd_zusuario TYPE REF TO if_wd_context_node.
  DATA lo_el_zusuario TYPE REF TO if_wd_context_element.
```

```
  DATA ls_zusuario TYPE wd_this->element_zusuario.
  DATA lv_usuario LIKE ls_zusuario-usuario.
  REFRESH it_gestion.

  SELECT * FROM zwda_cliente AS cliente
  INNER JOIN zhistorial AS historial
    ON cliente~identificador  =  historial~identificador
    AND cliente~tramite       =  historial~tramit
    AND cliente~circuito      =  historial~circuit
  INTO CORRESPONDING FIELDS OF TABLE it_historial
  WHERE circuit = circuito
  AND  rechazo NE 'X'
  AND  ( historial~resultat EQ space ).

* filtrar los trámites por rol.
  lo_nd_zusuario = wd_context->get_child_node( name = wd_this-
>wdctx_zusuario ).
*   get element via lead selection
  lo_el_zusuario = lo_nd_zusuario->get_element(  ).
*   get single attribute
  lo_el_zusuario->get_attribute(
    EXPORTING
      name =  `USUARIO`
    IMPORTING
      value = lv_usuario ).
  CALL FUNCTION 'CONVERSION_EXIT_ALPHA_INPUT'
    EXPORTING
      input  = lv_usuario
    IMPORTING
      output = lv_usuario.
  DATA l_rol TYPE char5.
  SELECT SINGLE rol FROM zusuarios  INTO l_rol
    WHERE usuario = lv_usuario.
  IF l_rol NE 'CLI'.
    IF l_rol NE '*'.
      DELETE it_historial WHERE tramit NE l_rol.
    ENDIF.
  ELSE.
    DELETE it_historial WHERE identificador NE lv_usuario.
  ENDIF.

  CLEAR  wa_gestion.
  DATA l_nombre TYPE string.
  LOOP AT  it_historial INTO wa_historial.
    SELECT SINGLE * FROM  zhistorial INTO wa_historial_aux
    WHERE identificador = wa_historial-identificador
    AND resultat = 'KO'.
    IF sy-subrc = 0.
      IF wa_historial_aux-tramit = 'RESCO'.
        wa_gestion-resco = '@02@'.
      ELSEIF wa_historial_aux-tramit = 'RESFI'.
        wa_gestion-resfi = '@02@'.
      ELSEIF wa_historial_aux-tramit = 'RESVE'.
        wa_gestion-resve = '@02@'.
      ENDIF.
    ENDIF.

    CASE wa_historial-tramit.
      WHEN 'CLI'.
        wa_gestion-cli     = '@A0@'.
      WHEN 'RESCO'.
        wa_gestion-resco   = '@A0@'.
        wa_gestion-cli     = '@08@'.
      WHEN 'RESFI'.
        wa_gestion-resco   = '@08@'.
        wa_gestion-cli     = '@08@'.
        wa_gestion-resfi   = '@A0@'.
```

```
      WHEN 'RESVE'.
        wa_gestion-resfi   = '@08@'.
        wa_gestion-cli     = '@08@'.
        wa_gestion-resco   = '@08@'.
        wa_gestion-resve   = '@A0@'.
      WHEN 'ALSAP'.
        wa_gestion-alsap   = '@4A@'.
        wa_gestion-resve   = '@08@'.
        wa_gestion-cli     = '@08@'.
        wa_gestion-resco   = '@08@'.
        wa_gestion-resfi   = '@08@'.
    ENDCASE.
    CALL FUNCTION 'CONVERSION_EXIT_ALPHA_OUTPUT'
      EXPORTING
        input  = wa_historial-identificador
      IMPORTING
        output = wa_gestion-identificador.
    CONCATENATE wa_gestion-identificador
    wa_historial-nombre
    INTO wa_gestion-identificador
    SEPARATED BY space.
    IF wa_historial-altasap = 'X'.
      wa_gestion-alsap   = '@5Y@'.
      wa_gestion-resve   = '@08@'.
      wa_gestion-cli     = '@08@'.
      wa_gestion-resco   = '@08@'.
      wa_gestion-resfi   = '@08@'.
    ENDIF.
    APPEND wa_gestion TO it_gestion.
    CLEAR  wa_gestion.
  ENDLOOP.
ENDMETHOD.
```

VALIDAR_ROL_TRAMITE

Comprobamos que el usuario tiene un rol que le permite realizar la operación.

```
METHOD validar_rol_tramite.
** rol usuario
  DATA lv_rol TYPE char5.
  DATA lo_nd_zusuario TYPE REF TO if_wd_context_node.
  DATA lo_el_zusuario TYPE REF TO if_wd_context_element.
  DATA ls_zusuario TYPE wd_this->element_zusuario.
  DATA lv_usuario LIKE ls_zusuario-usuario.
*   navigate from <CONTEXT> to <ZUSUARIO> via lead selection
  lo_nd_zusuario = wd_context->get_child_node( name = wd_this-
>wdctx_zusuario ).
*   get element via lead selection
  lo_el_zusuario = lo_nd_zusuario->get_element(  ).
*   get single attribute
  lo_el_zusuario->get_attribute(
    EXPORTING
      name =  `USUARIO`
    IMPORTING
      value = lv_usuario ).
  CALL FUNCTION 'CONVERSION_EXIT_ALPHA_INPUT'
    EXPORTING
      input  = lv_usuario
    IMPORTING
      output = lv_usuario.

  SELECT SINGLE rol FROM zusuarios INTO lv_rol
  WHERE usuario = lv_usuario.
* tramites asignados al rol
  DATA lit_roles TYPE STANDARD TABLE OF zwda_roles.
  SELECT  * FROM zwda_roles INTO TABLE lit_roles
  WHERE rol = lv_rol.
```

```
* comprobar si el rol tiene asignado el tramite actual
  DATA wa_tramites TYPE ztramites.
  READ TABLE lit_roles
  WITH KEY tramite =  tramite
  INTO wa_tramites.
if sy-subrc = 0.
  correcto = 'X'.
endif.
ENDMETHOD.
```

RETROCEDER_TRAMITE

La solicitud de cliente retrocede un trámite en función de la ruta, el circuito activo en el sistema y se registra en el historial.

```
METHOD retroceder_tramite.
  DATA: wa_historial TYPE zhistorial,
        wa_zbloqueos TYPE zbloqueos,
        l_prevtramite TYPE char5,
        l_maxnumord TYPE numc10,
        l_identificador TYPE char20.
  CALL FUNCTION 'CONVERSION_EXIT_ALPHA_INPUT'
    EXPORTING
      input  = identificador
    IMPORTING
      output = l_identificador.
*Comprobar que el proyecto  no esta bloqueado
  SELECT SINGLE * FROM zbloqueos
  INTO wa_zbloqueos WHERE identificador = l_identificador
  AND circuito = circuito AND tramite = tramite.
* comprobar que el proyecto no esta en tramite alta sap
  SELECT SINGLE * FROM zhistorial
  INTO wa_historia WHERE identificador = l_identificador
  AND   tramit         = 'ALSAP' AND   circuit         = circuito.
  IF ( wa_zbloqueos-bloq = ' ' AND sy-subrc = 4 ).
* Obtener la orden a actualizar
    SELECT MAX( numord ) INTO l_maxnumord
    FROM zhistorial
    WHERE identificador = l_identificador
    AND   tramit        = tramite
    AND   circuit       = circuito.
    SELECT SINGLE * FROM zhistorial
    INTO wa_historial
    WHERE identificador = l_identificador
    AND   tramit  = tramite
    AND   numord  = l_maxnumord
    AND   circuit = circuito.
* si existe la orden a actualizar
    IF sy-subrc = 0.
* actualizar historial
      wa_historial-datafi        = sy-datum.
      wa_historial-resultat      = 'KO'.
      wa_historial-userfin       = sy-uname.
      MODIFY zhistorial FROM wa_historial.
      COMMIT WORK.
    ENDIF.
* obtener previo tramite
    SELECT SINGLE prevtramit FROM zruta INTO l_prevtramite
    WHERE circuito = circuito AND tramite    = tramite
    AND resultado  = ' '.
* obtener el numero de orden del prevtramite
    SELECT MAX( numord ) INTO l_maxnumord
    FROM zhistorial WHERE identificador = l_identificador
    AND   tramit        = l_prevtramite AND  circuit = circuito.
```

```
* nuevo registro
    wa_historial-identificador = l_identificador.
    wa_historial-tramit = l_prevtramite.
    wa_historial-circuit = circuito.
    wa_historial-numord   = l_maxnumord + 1.
    wa_historial-resultat = ' '.
    wa_historial-datafi   = space.
    wa_historial-userfin  = space.
    INSERT into zhistorial values wa_historial.
    COMMIT WORK.
* actualizar tramite cliente.
    UPDATE zwda_cliente SET tramite         = l_prevtramite
    WHERE identificador = l_identificador.
    COMMIT WORK.
  ENDIF.
ENDMETHOD.
```

OB_CIRCUIT_TRAMITS

Obtenemos el circuito activo en sistema y cuál es el primer y último trámite que deben ejecutarse.

```
METHOD ob_circuit_tramits.
* recuperem el circuit actiu i el primer tramit i l'ultim
  SELECT SINGLE circuito INTO circuit
         FROM  zcircuito WHERE active = 'X'.

  SELECT SINGLE tramite INTO  tramit_ini
  FROM ztramites WHERE primtram = 'X'
  AND circuito = circuit.

  SELECT SINGLE sigtramit INTO   sigtramit
  FROM zruta WHERE tramite = tramit_ini
  AND circuito = circuit.

  SELECT SINGLE tramite  INTO  tramit_fin
  FROM ztramites WHERE ultram = 'X'
  AND circuito = circuit.

ENDMETHOD.
```

GENERAR_CLIENTE

Genera el cliente en SAP bajo la organización, área de ventas, canal de distribución determinados.

```
METHOD generar_cliente.

  DATA wa_cliente  TYPE zwda_cliente, l_error TYPE sy-subrc.
  DATA lit_messtab TYPE STANDARD TABLE OF  bdcmsgcoll.
  DATA wa_messtab  TYPE bdcmsgcoll, l_nombre TYPE char40.
  DATA wa_zhistorial TYPE zhistorial, lsyst TYPE syst.
  DATA lit_gestion TYPE STANDARD TABLE OF zstructalv.
  DATA lo_nd_zstruc_alv_cliente TYPE REF TO if_wd_context_node.
  DATA lo_el_zstruc_alv_cliente TYPE REF TO if_wd_context_element.
  DATA ls_zstruc_alv_cliente TYPE wd_this->element_zstruc_alv_cliente.
* get message manager
  DATA lo_api_controller     TYPE REF TO if_wd_controller.
  DATA lo_message_manager    TYPE REF TO if_wd_message_manager.
  lo_api_controller ?= wd_this->wd_get_api( ).
  CALL METHOD lo_api_controller->get_message_manager
    RECEIVING
```

```
      message_manager = lo_message_manager.
* recuperar datos cliente
  SELECT SINGLE * FROM zwda_cliente
  INTO wa_cliente WHERE identificador = identificador .
* comprobar que el cliente no está dado de alta en el sistema
  IF wa_cliente-altasap NE 'X'.
* comprobar si el cliente ha pasado la tramitación
    SELECT SINGLE * FROM zhistorial
    INTO  wa_zhistorial WHERE identificador = identificador
    AND   tramit        = 'ALSAP' AND   circuit       = wa_cliente-circuito.
    IF sy-subrc = 0.
      CALL FUNCTION 'ZWDA_CLIENTE_AUX'
       EXPORTING
*   BUKRS_001           = '1000'  "Sociedad
*   VKORG_002           = '1000'  "Area de ventas
*   VTWEG_003           = '10'    "Sector
*   SPART_004           = '00'    "canal
*   KTOKD_005           = 'ZINT'  "grupo de cuentas
      wa_cliente    =  wa_cliente
       IMPORTING
         subrc              = l_error
       TABLES
         messtab            = lit_messtab.
      READ TABLE lit_messtab WITH KEY msgid = 'F2' msgnr = 174
      msgspra = sy-langu and msgtyp  = 'S' INTO wa_messtab.
      IF sy-subrc = 0.
* actualizar zwda_cliente-altasap
        UPDATE zwda_cliente
        SET altasap = 'X' WHERE identificador = identificador.
        IF sy-subrc = 0.
* actualizar tabla zusuarios con el identificador de cliente
         UPDATE zusuarios
          SET cliente = wa_messtab-msgv1 WHERE usuario = identificador.
         IF sy-subrc = 0.
* enviar mail usuario
           wd_this->enviar_mail(
             identificador = identificador
               mail =  wa_cliente-mail).
           wd_this->rellenar_clientes_alv(
             CHANGING
               circuito   =   wa_cliente-circuito
               it_gestion =   lit_gestion ).
* linkar tabla
           lo_nd_zstruc_alv_cliente = wd_context-
>get_child_node( name = wd_this->wdctx_zstruc_alv_cliente ).
* @TODO handle not set lead selection
           IF NOT lo_nd_zstruc_alv_cliente IS INITIAL.
             CALL METHOD lo_nd_zstruc_alv_cliente->bind_table
               EXPORTING
                 new_items = lit_gestion.
           ENDIF.
*       report message
           lsyst-msgv1 = wa_messtab-msgv1.
           lsyst-msgv2 = wa_messtab-msgv2.
           lsyst-msgv3 = wa_messtab-msgv3.
           CALL METHOD lo_message_manager->report_t100_message
             EXPORTING
               msgid = wa_messtab-msgid
               msgno = '174'
               msgty = 'S'
               p1    = lsyst-msgv1
               p2    = lsyst-msgv2
               p3    = lsyst-msgv3..
         ENDIF.
        ENDIF.
      ENDIF.
*  el cliente no ha pasado la tramitación
```

```
    ELSE.
      CALL METHOD lo_message_manager->report_fatal_error_message
        EXPORTING
          message_text = text-006.
    ENDIF.
  ELSE.
    CALL METHOD lo_message_manager->report_warning
      EXPORTING
        message_text = text-007 '.
  ENDIF.
ENDMETHOD.
```

ZWDA_CLIENTE

Genera el cliente en SAP bajo la organización, área de ventas, canal de distribución determinados.

```
FUNCTION zwda_cliente.
  subrc = 0.

  PERFORM bdc_nodata      USING nodata.

  PERFORM open_group      USING group user keep holddate ctu.

  PERFORM bdc_dynpro      USING 'SAPMF02D' '0100'.
  PERFORM bdc_field       USING 'BDC_CURSOR'
                                'RF02D-KTOKD'.
  PERFORM bdc_field       USING 'BDC_OKCODE'
                                '/00'.
  PERFORM bdc_field       USING 'RF02D-BUKRS'
                                 bukrs_001.
  PERFORM bdc_field       USING 'RF02D-VKORG'
                                 vkorg_002.
  PERFORM bdc_field       USING 'RF02D-VTWEG'
                                 vtweg_003.
  PERFORM bdc_field       USING 'RF02D-SPART'
                                 spart_004.
  PERFORM bdc_field       USING 'RF02D-KTOKD'
                                 ktokd_005.

  PERFORM bdc_dynpro      USING 'SAPMF02D' '0110'.
  PERFORM bdc_field       USING 'BDC_CURSOR'
                                'KNA1-SPRAS'.
  PERFORM bdc_field       USING 'BDC_OKCODE'
                                '/00'.
  PERFORM bdc_field       USING 'KNA1-NAME1'
                                 wa_cliente-nombre.
  PERFORM bdc_field       USING 'KNA1-LAND1'
                                 wa_cliente-pais.
  PERFORM bdc_field       USING 'KNA1-SPRAS'
                                 wa_cliente-idioma.
  PERFORM bdc_field       USING 'KNA1-STRAS'
                                  wa_cliente-calle.
  PERFORM bdc_field       USING 'KNA1-ORT01'
                                 wa_cliente-poblacion.
  PERFORM bdc_field       USING 'KNA1-PSTLZ'
                                 wa_cliente-codpostal.
  PERFORM bdc_field       USING 'KNA1-REGIO'
                                 wa_cliente-ciudad.
  PERFORM bdc_field       USING 'KNA1-TELF1'
                                 wa_cliente-telefono.
  PERFORM bdc_field       USING 'KNA1-KNURL'
                                 wa_cliente-mail.
```

```
PERFORM bdc_dynpro      USING 'SAPMF02D' '0120'.
PERFORM bdc_field       USING 'BDC_CURSOR'
                              'KNA1-STCD1'.
PERFORM bdc_field       USING 'KNA1-STCD1'
                             wa_cliente-DNI.
PERFORM bdc_field       USING 'BDC_OKCODE'
                              '/00'.

PERFORM bdc_dynpro      USING 'SAPMF02D' '0125'.
PERFORM bdc_field       USING 'BDC_CURSOR'
                              'KNA1-NIELS'.
PERFORM bdc_field       USING 'BDC_OKCODE'
                              '/00'.

PERFORM bdc_dynpro      USING 'SAPMF02D' '0130'.
PERFORM bdc_field       USING 'BDC_CURSOR'
                              'KNBK-BKONT(01)'.
PERFORM bdc_field       USING 'BDC_OKCODE'
                              'ENTR'.
PERFORM bdc_field       USING 'KNBK-BANKS(01)'
                               wa_cliente-clavepais.
PERFORM bdc_field       USING 'KNBK-BANKL(01)'
                               wa_cliente-banco.
PERFORM bdc_field       USING 'KNBK-BANKN(01)'
                               wa_cliente-cuenta.
PERFORM bdc_field       USING 'KNBK-BKONT(01)'
                               wa_cliente-clavecontrol.

PERFORM bdc_dynpro      USING 'SAPMF02D' '0130'.
PERFORM bdc_field       USING 'BDC_CURSOR'
                              'KNBK-BANKS(01)'.
PERFORM bdc_field       USING 'BDC_OKCODE'
                              '=VW'.

PERFORM bdc_dynpro      USING 'SAPMF02D' '0340'.
PERFORM bdc_field       USING 'BDC_CURSOR'
                              'RF02D-KUNNR'.
PERFORM bdc_field       USING 'BDC_OKCODE'
                              '=VW'.

PERFORM bdc_dynpro   USING 'SAPMF02D' '0370'.
PERFORM bdc_field    USING 'BDC_CURSOR'
                           'RF02D-KUNNR'.
PERFORM bdc_field    USING 'BDC_OKCODE'
                           '=VW'.
PERFORM bdc_field    USING 'KNA1-CIVVE'
                           civve_013.

PERFORM bdc_dynpro   USING 'SAPMF02D' '0360'.
PERFORM bdc_field    USING 'BDC_CURSOR'
                           'KNVK-NAMEV(01)'.
PERFORM bdc_field    USING 'BDC_OKCODE'
                           '=VW'.


PERFORM bdc_dynpro   USING 'SAPMF02D' '0210'.
PERFORM bdc_field    USING 'BDC_CURSOR'
                           'KNB1-AKONT'.
PERFORM bdc_field    USING 'BDC_OKCODE'
                               '=VW'.

PERFORM bdc_dynpro   USING 'SAPMF02D' '0215'.
PERFORM bdc_field    USING 'BDC_CURSOR'
                           'KNB1-ZTERM'.
PERFORM bdc_field    USING 'BDC_OKCODE'
                           '=VW'.
```

```
PERFORM bdc_dynpro   USING 'SAPMF02D' '0220'.
PERFORM bdc_field    USING 'BDC_CURSOR'
                           'KNB5-MAHNA'.
PERFORM bdc_field    USING 'BDC_OKCODE'
                           '=VW'.

PERFORM bdc_dynpro   USING 'SAPMF02D' '0230'.
PERFORM bdc_field    USING 'BDC_CURSOR'
                           'KNB1-VRSNR'.
PERFORM bdc_field    USING 'BDC_OKCODE'
                           '=VW'.

PERFORM bdc_dynpro      USING 'SAPMF02D' '0610'.
PERFORM bdc_field       USING 'BDC_OKCODE'
                           '=VW'.
PERFORM bdc_field       USING 'BDC_CURSOR'
                           'RF02D-KUNNR'.

PERFORM bdc_dynpro      USING 'SAPMF02D' '0310'.
PERFORM bdc_field       USING 'BDC_CURSOR'
                           'KNVV-VERSG'.
PERFORM bdc_field       USING 'BDC_OKCODE'
                           '=VW'.
PERFORM bdc_field       USING 'KNVV-AWAHR'
                            awahr_014.
PERFORM bdc_field       USING 'KNVV-WAERS'
                            waers_015.
PERFORM bdc_field       USING 'KNVV-KALKS'
                            kalks_016.
PERFORM bdc_field       USING 'KNVV-VERSG'
                            versg_017.

PERFORM bdc_dynpro      USING 'SAPMF02D' '0315'.
PERFORM bdc_field       USING 'BDC_CURSOR'
                           'KNVV-VSBED'.
PERFORM bdc_field       USING 'BDC_OKCODE'
                           '=VW'.
PERFORM bdc_field       USING 'KNVV-KZAZU'
                            kzazu_018.
PERFORM bdc_field       USING 'KNVV-VSBED'
                            vsbed_019.
PERFORM bdc_field       USING 'KNVV-ANTLF'
                            antlf_020.

PERFORM bdc_dynpro      USING 'SAPMF02D' '0320'.
PERFORM bdc_field       USING 'BDC_CURSOR'
                           'KNVV-PERFK'.
PERFORM bdc_field       USING 'BDC_OKCODE'
                           '=VW'.

PERFORM bdc_dynpro      USING 'SAPMF02D' '1350'.
PERFORM bdc_field       USING 'BDC_CURSOR'
                           'KNVI-TAXKD(04)'.
PERFORM bdc_field       USING 'BDC_OKCODE'
                           '=VW'.
PERFORM bdc_field       USING 'KNVI-TAXKD(01)'
                          taxkd_01_021.
PERFORM bdc_field       USING 'KNVI-TAXKD(02)'
                          taxkd_02_022.
PERFORM bdc_field       USING 'KNVI-TAXKD(03)'
                          taxkd_03_023.
PERFORM bdc_field       USING 'KNVI-TAXKD(04)'
                          taxkd_04_024.
PERFORM bdc_dynpro      USING 'SAPMF02D' '0324'.
PERFORM bdc_field       USING 'BDC_CURSOR'
                           'KNVP-PARVW(01)'.
```

```
  PERFORM bdc_field       USING 'BDC_OKCODE'
                                '=VW'.

  PERFORM bdc_transaction TABLES messtab
  USING                         'XD01' ctu mode update.
  PERFORM close_group USING     ctu.
ENDFUNCTION.


FORM BDC_NODATA USING P_NODATA.
  NODATA_CHARACTER = P_NODATA.
ENDFORM.

FORM OPEN_GROUP
     USING P_GROUP    LIKE APQI-GROUPID
           P_USER     LIKE APQI-USERID
           P_KEEP     LIKE APQI-QERASE
           P_HOLDDATE LIKE APQI-STARTDATE
           P_CTU      LIKE APQI-PUTACTIVE.

  IF P_CTU <> 'X'.
    CALL FUNCTION 'BDC_OPEN_GROUP'
         EXPORTING  CLIENT   = SY-MANDT
                    GROUP    = P_GROUP
                    USER     = P_USER
                    KEEP     = P_KEEP
                    HOLDDATE = P_HOLDDATE.
   ENDIF.
ENDFORM.

FORM BDC_DYNPRO USING PROGRAM DYNPRO.
  CLEAR BDCDATA.
  BDCDATA-PROGRAM  = PROGRAM.
  BDCDATA-DYNPRO   = DYNPRO.
  BDCDATA-DYNBEGIN = 'X'.
  APPEND BDCDATA.
ENDFORM.


FORM BDC_FIELD USING FNAM FVAL.
  IF fval <> NODATA_CHARACTER.
    CLEAR BDCDATA.
    BDCDATA-FNAM = FNAM.
    BDCDATA-FVAL = FVAL.
    APPEND BDCDATA.
  ENDIF.
ENDFORM.


FORM CLOSE_GROUP USING P_CTU LIKE APQI-PUTACTIVE.
  IF P_CTU <> 'X'.
* close batchinput group
    CALL FUNCTION 'BDC_CLOSE_GROUP'.
  ENDIF.
ENDFORM.

FORM BDC_TRANSACTION TABLES P_MESSTAB
                     USING  P_TCODE P_CTU P_MODE P_UPDATE.
DATA: L_SUBRC LIKE SY-SUBRC.

  IF P_CTU <> 'X'.
    CALL FUNCTION 'BDC_INSERT'
         EXPORTING  TCODE     = P_TCODE
         TABLES     DYNPROTAB = BDCDATA
         EXCEPTIONS OTHERS    = 1.
  ELSE.
```

```
    CALL TRANSACTION P_TCODE USING BDCDATA
                     MODE   P_MODE UPDATE P_UPDATE
                     MESSAGES INTO P_MESSTAB.
  ENDIF.
  L_SUBRC = SY-SUBRC.
  REFRESH BDCDATA.
  SY-SUBRC = L_SUBRC.
ENDFORM.
```

AVANZA_SIG_TRAMITE

La solicitud de cliente avanza un trámite en el circuito y se registra en el historial.

```
METHOD avanza_sig_tramite.
  DATA: wa_historial TYPE zhistorial, l_sigtramit TYPE char5,
        l_numord   TYPE numc10.
  DATA  l_tramite_fin TYPE char5.
  DATA: wa_zwda_cliente TYPE zwda_cliente, wa_zhistorial   TYPE zhistorial,
        wa_zbloqueos    TYPE zbloqueos, l_identificador TYPE char20.
  CALL FUNCTION 'CONVERSION_EXIT_ALPHA_INPUT'
    EXPORTING
      input  = identificador
    IMPORTING
      output = l_identificador.
* obtener el numero de orden
    SELECT MAX( numord ) INTO l_numord FROM zhistorial
    WHERE identificador = l_identificado AND   circuit       = circuito
    AND   tramit        = tramite.
* obtener el tramite a actualizar
    SELECT SINGLE * FROM zhistorial INTO wa_historial
    WHERE identificador = l_identificador AND   circuit       = circuito
    AND   tramit        = tramite         AND   numord        = l_numord.
*  si existe la solicitud
    IF sy-subrc = 0.
* actualizarlo
      wa_historial-datafi        = sy-datum.
      wa_historial-resultat      = 'OK'.
      wa_historial-userfin       = sy-uname.
      MODIFY zhistorial FROM wa_historial.
      COMMIT WORK.
* primer tramite
    ELSE.
      wa_historial-identificador = l_identificador.
      wa_historial-tramit        = tramite.
      wa_historial-numord        =  1.
      wa_historial-circuit       = circuito.
      wa_historial-dataini       = sy-datum.
      wa_historial-datafi        = sy-datum.
      wa_historial-resultat      = 'OK'.
      wa_historial-userfin       = sy-uname.
      INSERT into zhistorial values wa_historial.
      COMMIT WORK.
    ENDIF.
* obtener el tramite siguiente
    SELECT SINGLE sigtrami FROM zruta INTO l_sigtramit
    WHERE circuito = circuito AND   tramite  = tramite
    AND   resultado = 'X'.

* obtener el numero de orden
    SELECT MAX( numord ) INTO l_numord FROM zhistorial
    WHERE identificador = l_identificado
    AND   circuit       = circuito AND   tramit = l_sigtramit.
*nuevo tramite
    wa_historial-identificador = l_identificador.
    wa_historial-numord        = l_numord + 1.
    wa_historial-tramit        = l_sigtramit.
```

```
    wa_historial-circuit       = circuito.
    wa_historial-dataini       = sy-datum.
    wa_historial-datafi        = space.
    wa_historial-resultat      = ' '.
    wa_historial-userfin       = space.
    INSERT into zhistorial values wa_historial.
    COMMIT WORK.
* actualizar el tramite cliente
    UPDATE zwda_cliente SET tramite = l_sigtramit
    WHERE identificador = l_identificador.
    COMMIT WORK.
ENDMETHOD.
```

CREATE_ALV_HISTORIAL

crea el objeto historial de un cliente, mediante el historial se visualiza toda la información relacionada con los distintos trámites por los que ha pasado una solicitud de cliente.

```
METHOD create_alv_historial.
  DATA  lo_cmp_usage TYPE REF TO if_wd_component_usage.
  lo_cmp_usage =   wd_this->wd_cpuse_alv_historial( ).
  IF lo_cmp_usage->has_active_component( ) IS INITIAL.
    lo_cmp_usage->create_component( ).
  ENDIF.
  ref_interfacecontroller = wd_this->wd_cpifc_alv_historial( ).
  alv_historial = ref_interfacecontroller->get_model( ).
  alv_historial->if_salv_wd_table_settings~set_visible_row_count( '15' ).

  DATA: lr_column_settings TYPE REF TO if_salv_wd_column_settings.
  DATA: lr_column TYPE REF TO cl_salv_wd_column.
* Columnas
  lr_column_settings ?= alv_historial.
  lr_column = lr_column_settings->get_column( 'IDENTIFICADOR' ).
  lr_column->if_salv_wd_column_hierarchy~set_hierarchy_column( abap_true ).
  CALL METHOD lr_column_settings->delete_column
    EXPORTING
      id = 'NUMORD'.
  CALL METHOD lr_column_settings->delete_column
    EXPORTING
      id = 'CIRCUIT'.
  CALL METHOD lr_column_settings->delete_column
    EXPORTING
      id = 'MANDT'.
  CALL METHOD lr_column_settings->delete_column
    EXPORTING
      id = 'USERFIN'.
  DATA: lr_header_alv TYPE REF TO cl_salv_wd_column_header.
  lr_column = lr_column_settings->get_column( 'IDENTIFICADOR' ).
  lr_header_alv = lr_column->get_header( ).
  lr_header_alv->set_text( 'Nombre' ).
  lr_column = lr_column_settings->get_column( 'TRAMIT' ).
  lr_header_alv = lr_column->get_header( ).
  lr_header_alv->set_text( 'Tramite' ).
  lr_column = lr_column_settings->get_column( 'DATAINI' ).
  lr_header_alv = lr_column->get_header( ).
  lr_header_alv->set_text( 'Fecha inicial' ).
  lr_column = lr_column_settings->get_column( 'RESULTAT' ).
  lr_header_alv = lr_column->get_header( ).
  lr_header_alv->set_text( 'Resultado trámite' ).
  lr_column = lr_column_settings->get_column( 'DATAFI' ).
  lr_header_alv = lr_column->get_header( ).
  lr_header_alv->set_text( 'Fecha final' ).
```

```
* tabla
 DATA: lr_table_settings TYPE REF TO if_salv_wd_table_settings.
 lr_table_settings ?= alv_historial.
  CALL METHOD lr_table_settings->set_display_type
    EXPORTING
      value = if_salv_wd_c_table_settings=>display_type_hierarchy.
 alv_historialif_salv_wd_std_functions~set_filter_complex_allowed( abap_false
  alv_historial-
>if_salv_wd_std_functions~set_filter_filterline_allowed( abap_false ).
  alv_historial->if_salv_wd_std_functions~set_view_list_allowed( abap_false ).
  alv_historial->if_salv_wd_std_functions~set_export_allowed( abap_false ).
  alv_historial->if_salv_wd_std_functions~set_pdf_allowed( abap_false ).
  alv_historial-
>if_salv_wd_table_settings~set_fixed_table_layout( ).  DATA: lr_image  TYPE RE
F TO cl_salv_wd_uie_image.
  lr_column_settings ?= alv_HISTORIAL.
  CREATE OBJECT lr_image.
  lr_column = lr_column_settings->get_column( 'RESULTAT' ).
  lr_image->set_source_fieldname( 'RESULTAT').
  lr_column->set_cell_editor( lr_image ).
ENDMETHOD.
```

CREATE_ALV_COMPRAR

crea el objeto pedido y modifica algunos parámetros relacionados con la visualización por pantalla.

```
METHOD create_alv_comprar .
  DATA  lo_cmp_usage TYPE REF TO if_wd_component_usage.
  DATA: lr_column_settings TYPE REF TO if_salv_wd_column_settings.
  DATA: lr_column TYPE REF TO cl_salv_wd_column, l_descrip TYPE string
  DATA lr_header_alv TYPE REF TO cl_salv_wd_column_header.
  lo_cmp_usage =   wd_this->wd_cpuse_alv_comprar( ).
  IF lo_cmp_usage->has_active_component( ) IS INITIAL.
    lo_cmp_usage->create_component( ).
  ENDIF.
  ref_interfacecontroller = wd_this->wd_cpifc_alv_comprar( ).
  alv_comprar = ref_interfacecontroller->get_model( ).
  alv_comprar-
>if_salv_wd_table_settings~set_visible_row_count( '10' ).
  alv_comprar-
>if_salv_wd_std_functions~set_dialog_settings_allowed( abap_false ).
  alv_comprar-
>if_salv_wd_std_functions~set_display_settings_allowed( abap_false ).
  alv_comprar-
>if_salv_wd_std_functions~set_filter_complex_allowed( abap_false ).
  alv_comprar-
>if_salv_wd_std_functions~set_filter_filterline_allowed( abap_false ).
  alv_comprar->if_salv_wd_std_functions~set_view_list_allowed( abap_false ).
  alv_comprar->if_salv_wd_std_functions~set_export_allowed( abap_false ).
  alv_comprar->if_salv_wd_std_functions~set_pdf_allowed( abap_false ).
  alv_comprar->if_salv_wd_table_settings~set_fixed_table_layout( ).
  lr_column_settings ?= alv_comprar.
  CALL METHOD lr_column_settings->delete_column
    EXPORTING
      id = 'PEDIDO'.
  CALL METHOD lr_column_settings->delete_column
    EXPORTING
      id = 'FECHA'.
  CALL METHOD lr_column_settings->delete_column
    EXPORTING
      id = 'PESO'.
```

```
  CALL METHOD lr_column_settings->delete_column
    EXPORTING
      id = 'UNIDAD'.
    CALL METHOD lr_column_settings->delete_column
    EXPORTING
      id = 'STATUS'.
  wd_this->ordenar_colum_alvcomprar(
    alv_comprar =   alv_comprar                    " ref to cl_salv_wd_config_
table ).
  l_descrip = 'Precio unitario'.
  lr_column = lr_column_settings->get_column( 'PRECIO_UNI' ).
  lr_header_alv = lr_column->get_header( ).
  lr_header_alv->set_text( l_descrip ).
ENDMETHOD.
```

CREATE_ALV_PEDIDOS

Crea el objeto listado de los pedidos que tiene un cliente determinado pendientes de entregar.

```
METHOD create_alv_pedidos.
* tabla
  DATA: lr_table_settings TYPE REF TO if_salv_wd_table_settings.
  lr_table_settings ?= alv_pedidos.
  DATA  lo_cmp_usage TYPE REF TO if_wd_component_usage.
  lo_cmp_usage =   wd_this->wd_cpuse_alv_pedidos( ).
  IF lo_cmp_usage->has_active_component( ) IS INITIAL.
    lo_cmp_usage->create_component( ).
  ENDIF.
  ref_interfacecontroller = wd_this->wd_cpifc_alv_pedidos( ).
  alv_pedidos = ref_interfacecontroller->get_model( ).
  alv_pedidos->if_salv_wd_table_settings~set_visible_row_count( '10' ).
  alv_pedidos->if_salv_wd_std_functions~set_view_list_allowed( abap_false ).
  alv_pedidos->if_salv_wd_std_functions~set_export_allowed( abap_false ).
  alv_pedidos->if_salv_wd_std_functions~set_pdf_allowed( abap_false ).
  alv_pedidos->if_salv_wd_table_settings~set_fixed_table_layout( ).
  CALL METHOD lr_table_settings->set_display_type
    EXPORTING
      value = if_salv_wd_c_table_settings=>display_type_hierarchy.
* columnas
  DATA: lr_column_settings TYPE REF TO if_salv_wd_column_settings.
  DATA: lr_column TYPE REF TO cl_salv_wd_column.

  lr_column_settings ?= alv_pedidos.
  lr_column = lr_column_settings->get_column( 'PEDIDO' ).
  lr_column->if_salv_wd_column_hierarchy~set_hierarchy_column( abap_true ).
  CALL METHOD lr_column_settings->delete_column
    EXPORTING
      id = 'STATUS'.
ENDMETHOD.
```

AGREGAR_TOTALES_ALV_COMPRAR

Modifica el objeto pedido para que muestre el total del pedido.

```
METHOD agregar_totales_alvcomprar.
 DATA: lr_field TYPE REF TO cl_salv_wd_field.
* For total calculations...
  alv_comprar->if_salv_wd_std_functions~set_aggregation_allowed( abap_true ).
  lr_field = alv_comprar->if_salv_wd_field_settings~get_field( 'PRECIO_TOT' ).
  lr_field-
>if_salv_wd_aggr~create_aggr_rule( aggregation_type = if_salv_wd_c_aggregation
```

```
=>aggrtype_total ).
ENDMETHOD.
```

AGREGAR_TOTALES_ALVPEDIDOS

Modifica el objeto listado de pedidos pendientes para que muestre el total de todos los pedidos y el total por pedido.

```
METHOD agregar_totales_alvpedidos.
  DATA: lr_field TYPE REF TO cl_salv_wd_field.
* For total calculations...
  alv_pedidos->if_salv_wd_std_functions~set_aggregation_allowed( abap_true ).
  lr_field = alv_pedidos->if_salv_wd_field_settings~get_field( 'PRECIO_UNI' ).
  lr_field-
>if_salv_wd_aggr~create_aggr_rule( aggregation_type = if_salv_wd_c_aggregation
=>aggrtype_total ).
ENDMETHOD.
```

CREATE_FUNCTIONS_ALV_COMPRAR

Este método crea las funciones que se podrán realizar desde el apartado realizar pedidos y son las siguientes:

- Crear el pedido en Sap.
- eliminar un producto del pedido.

```
METHOD CREATE_FUNCTIONS_ALV_COMPRAR.

  DATA lr_functions TYPE REF TO if_salv_wd_function_settings.
  lr_functions  = ref_interfacecontroller->get_model( ).
  DATA: lr_function TYPE REF TO cl_salv_wd_function.
  DATA: lr_button   TYPE REF TO cl_salv_wd_fe_button.
  DATA: lr_image    TYPE REF TO cl_salv_wd_uie_image.

  lr_function = lr_functions->create_function( 'TRANSFERIR' ).
  CREATE OBJECT lr_button.
  lr_button->set_text( 'Realizar pedido' ).
  CALL METHOD lr_button->set_image_source( 'ICON_IMPORT_TRANSPORT_REQUEST' ).
  lr_function->set_editor( lr_button ).
  lr_function = lr_functions->create_function( 'BORRAR' ).
  CREATE OBJECT lr_button.
  lr_button->set_text( 'Eliminar' ).
  CALL METHOD lr_button->set_image_source( 'ICON_DELETE' ).
  lr_function->set_editor( lr_button ).

ENDMETHOD.
```

CREATE_TITLE_ALV_COMPRAR

Crea el título para el objeto pedido.

```
METHOD CREATE_TITLE_ALV_COMPRAR.

  DATA: lr_header TYPE REF TO cl_salv_wd_header.
  DATA: lr_table_settings TYPE REF TO if_salv_wd_table_settings.
  lr_table_settings ?= alv_comprar.

  lr_header = lr_table_settings->get_header( ).
  lr_header->set_text( 'text-003').

ENDMETHOD.
```

CREATE_TITLE_ALV_PEDIDOS

crea el título para el objeto listado de pedidos.

```
METHOD CREATE_TITLE_ALV_PEDIDOS.

  DATA: lr_header TYPE REF TO cl_salv_wd_header.
  DATA: lr_table_settings TYPE REF TO if_salv_wd_table_settings.

  lr_table_settings ?= alv_PEDIDOS.
  lr_header = lr_table_settings->get_header( ).
  lr_header->set_text( 'text-004').

ENDMETHOD.
```

ORDENAR_COLUMNAS_ALVCOMPRAR

Poner la descripción del producto a comprar en primera posición seguido del precio correspondiente.

```
METHOD ordenar_colum_alvcomprar.

  DATA: lr_column_settings TYPE REF TO if_salv_wd_column_settings.
  DATA l_position TYPE i, lr_column TYPE REF TO cl_salv_wd_column.

  lr_column_settings ?= alv_comprar.
  l_position = 1.
  CALL METHOD lr_column_settings->get_column
    EXPORTING
      id    = 'DESCRIP'
    RECEIVING
      value = lr_column.
  lr_column->set_position( l_position ).
   l_position = 4.
  CALL METHOD lr_column_settings->get_column
    EXPORTING
      id    = 'PRECIO_UNI'
    RECEIVING
      value = lr_column.
  lr_column->set_position( l_position ).

ENDMETHOD
```

7.2 El controlador autentificación

7.2.1 Contexto:

En el controlador autentificación el contexto está formado por las clases que muestra la figura

Clase	Estructura
Usuario: los datos del usuario definidos en la especificación	Zusuario_change
	Zusuario_forget

Tabla 8: clases utilizadas

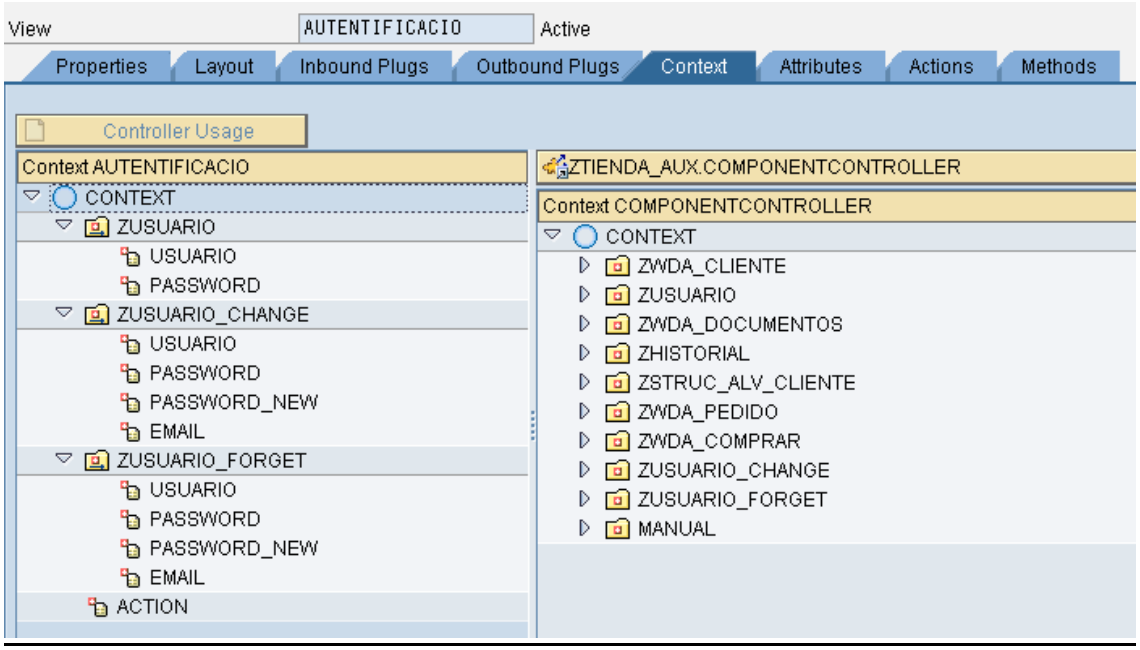


Figura 5: Contexto del componente autentificación

7.2.2 Vista:

En este apartado se muestra la interfaz de usuario para la autentificación.

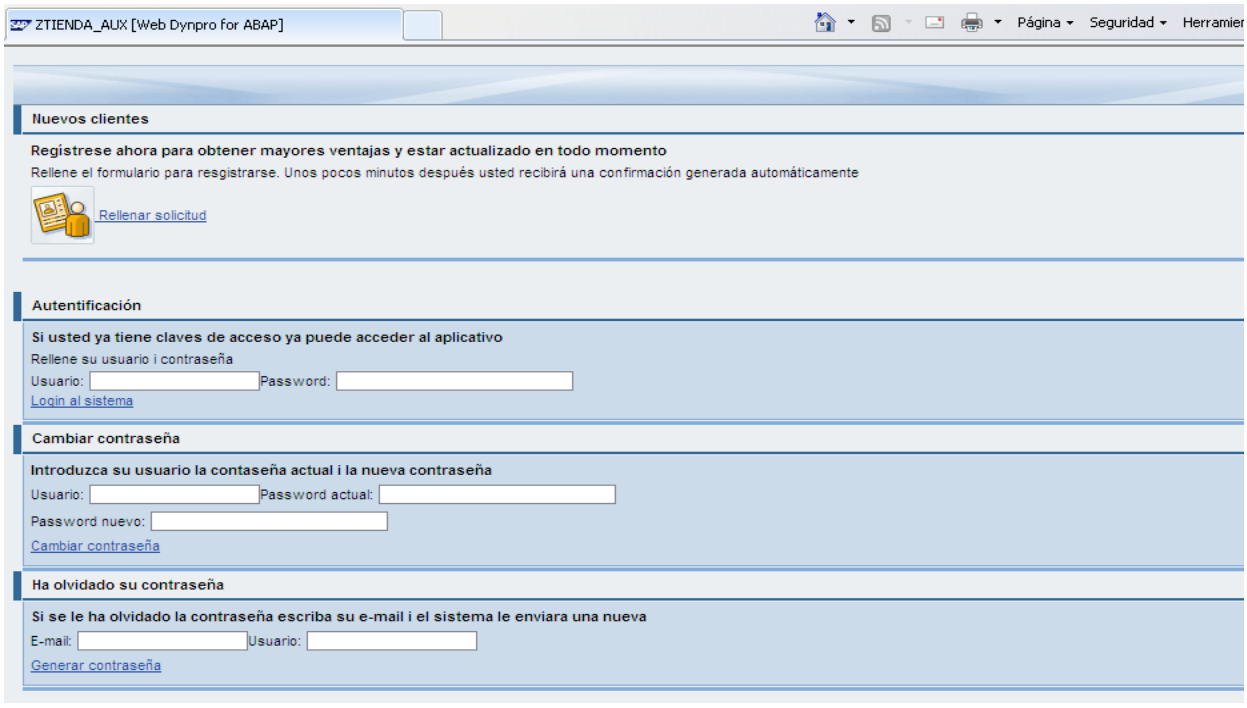


Figura 6: interfaz de usuario para autentificación

7.4.2.1 Conector entrada: se ha creado un conector de entrada a la vista autentificación mediante el cual se accede.

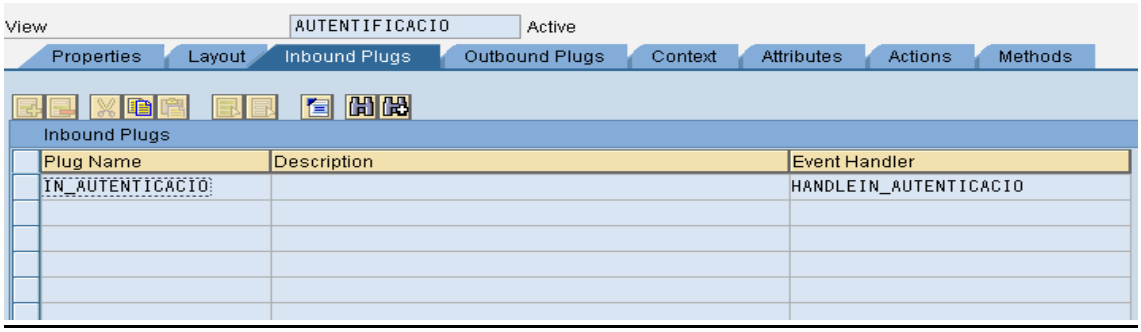


Figura 7: conector de entrada

7.4.2.2 Conector salida: se han creado dos conectores de salida con destino a rellenar el formulario y entrada a la gestión de clientes y de pedidos si la autentificación es correcta.

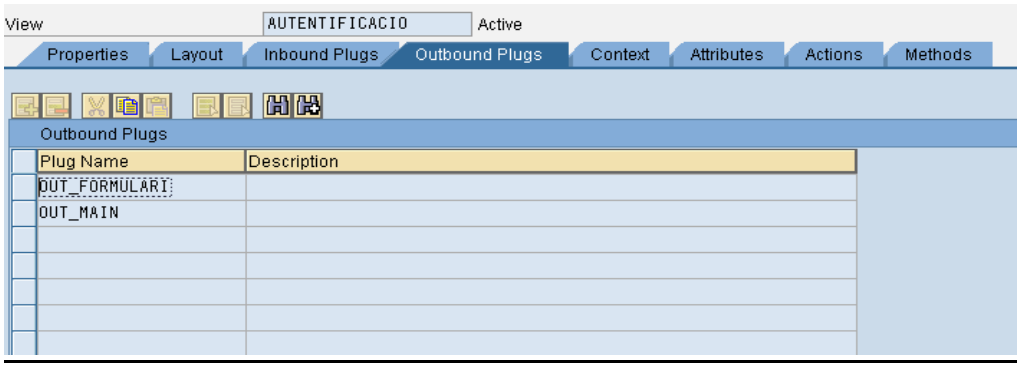


Figura 8: conectores de salida

7.2.3 Métodos

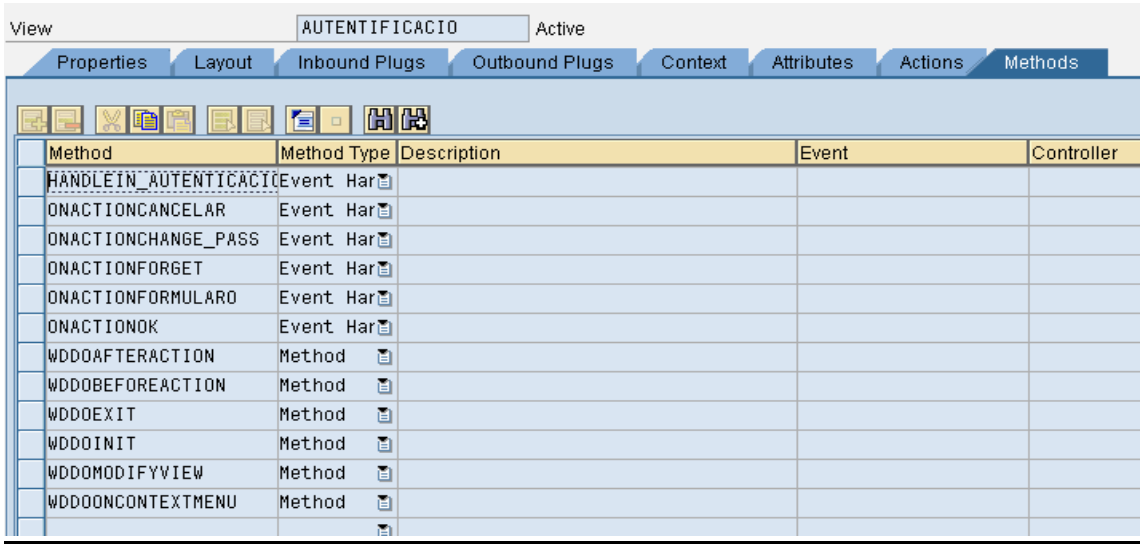


Figura 9: Métodos del componente autentificación

ONACTIONHANGE_PASS

Este método ejecuta el cambio de contraseña

```
METHOD onactionchange_pass.

  DATA lo_componentcontroller TYPE REF TO ig_componentcontroller .
  lo_componentcontroller =   wd_this->get_componentcontroller_ctr( ).

  lo_componentcontroller->change_password(
      i_view = 'AUTENTIFICACIO'          ).

ENDMETHOD.
```

ONACTIONFORGET

Este método ejecuta la generación de una nueva contraseña

```
METHOD onactionforget.

  DATA lo_componentcontroller TYPE REF TO ig_componentcontroller .
  lo_componentcontroller =   wd_this->get_componentcontroller_ctr( ).
  lo_componentcontroller->forget_password().

ENDMETHOD.
```

FORGET_PASSWORD

El sistema proporciona una nueva contraseña a un usuario que la olvido mediante email.

```
METHOD forget_password.
  DATA lpass_out TYPE wt_authval, lpass_in  TYPE wt_authval.
  DATA lo_nd_zusuario_forget TYPE REF TO if_wd_context_node.
  DATA lo_el_zusuario_forget TYPE REF TO if_wd_context_element.
  DATA ls_zusuario_forget TYPE wd_this->element_zusuario_forget.
  DATA lv_email LIKE ls_zusuario_forget-email, lv_usuario LIKE zus.
  DATA wa_usuario TYPE zusuarios.
  lo_nd_zusuario_forget = wd_context->get_child_node( name = wd_this
>wdctx_zusuario_forget ).
  lo_el_zusuario_forget = lo_nd_zusuario_forget->get_element(  ).
  lo_el_zusuario_forget->get_attribute(
    EXPORTING
      name =  `EMAIL`
    IMPORTING
      value = lv_email ).
*  get single attribute
  lo_el_zusuario_forget->get_attribute(
    EXPORTING
      name =  `USUARIO`
    IMPORTING
      value = lv_usuario ).
  CALL FUNCTION 'CONVERSION_EXIT_ALPHA_INPUT'
    EXPORTING
      input  = lv_usuario
    IMPORTING
      output = lv_usuario.

* generar password
  lpass_in = lv_usuario.
```

```
  CALL FUNCTION 'IDWT_CIS_PASSWORD_HASH'
    EXPORTING
      l_password = lpass_in
    IMPORTING
      value      = lpass_out.
  CALL FUNCTION 'ZWDA_MAIL'
    EXPORTING
      i_usuario = lv_usuario
      i_pass    = lpass_out
      i_mail    = lv_email.

* modificar tabla zusuarios
  SELECT SINGLE * FROM zusuario INTO wa_usuario
  WHERE usuario = lv_usuario.
  IF sy-subrc = 0.
    wa_usuario-pass = lpass_out.
    MODIFY zusuarios FROM wa_usuario.
    IF sy-subrc = 0.
      DATA lo_api_controller     TYPE REF TO if_wd_controller.
      DATA lo_message_manager    TYPE REF TO if_wd_message_manager.
      lo_api_controller ?= wd_this->wd_get_api( ).
      CALL METHOD lo_api_controller->get_message_manager
        RECEIVING
          message_manager = lo_message_manager.
*       report message
      CALL METHOD lo_message_manager->report_success
        EXPORTING
          message_text         = text-005.
    ENDIF.
  ENDIF.
  IF NOT lo_nd_zusuario_forget IS INITIAL.
    CALL METHOD lo_nd_zusuario_forget->set_static_attributes_null
   ENDIF.
ENDMETHOD.
```

CHANGE_PASSWORD

El sistema permite cambiar al usuario su contraseña.

```
METHOD change_password.
  DATA wa_usuarios TYPE zusuarios.
  DATA lo_nd_zusuario_change TYPE REF TO if_wd_context_node.
  DATA lo_el_zusuario_change TYPE REF TO if_wd_context_element.
  DATA ls_zusuario_change TYPE wd_this->element_zusuario_change.
  DATA: lv_usuario LIKE ls_zusuario_change-usuario,
        lv_password LIKE ls_zusuario_change-password,
        lv_password_new LIKE ls_zusuario_change-password_new.
* navigate from <CONTEXT> to <ZUSUARIO_CHANGE> via lead selection
  lo_nd_zusuario_change = wd_context->get_child_node( name = wd_this-
>wdctx_zusuario_change ).
* get element via lead selection
  lo_el_zusuario_change = lo_nd_zusuario_change->get_element(  ).
* get single attribute
  lo_el_zusuario_change->get_attribute(
    EXPORTING
      name =  `USUARIO`
    IMPORTING
      value = lv_usuario ).
  CALL FUNCTION 'CONVERSION_EXIT_ALPHA_INPUT'
    EXPORTING
      input  = lv_usuario
    IMPORTING
      output = lv_usuario.
* get single attribute
  lo_el_zusuario_change->get_attribute(
    EXPORTING
```

```
      name =  `PASSWORD`
    IMPORTING
      value = lv_password ).
  SELECT SINGLE * FROM zusuarios INTO  wa_usuarios
  WHERE usuario = lv_usuari AND   pass    = lv_password.
  IF sy-subrc = 0.
* get single attribute
    lo_el_zusuario_change->get_attribute(
      EXPORTING
        name =  `PASSWORD_NEW`
      IMPORTING
        value = lv_password_new ).
    wa_usuarios-pass = lv_password_new.
    MODIFY zusuarios FROM wa_usuarios.
    IF sy-subrc = 0.
* get message manager
      DATA lo_api_controller     TYPE REF TO if_wd_controller.
      DATA lo_message_manager    TYPE REF TO if_wd_message_manager.
      lo_api_controller ?= wd_this->wd_get_api( ).
      CALL METHOD lo_api_controller->get_message_manager
        RECEIVING
          message_manager = lo_message_manager.
* report message
      CALL METHOD lo_message_manager->report_success
        EXPORTING
          message_text             = text-001.

    ENDIF.
  ENDIF.
  IF NOT lo_nd_zusuario_change IS INITIAL.
    CALL METHOD lo_nd_zusuario_change->set_static_attributes_null.
  ENDIF.
ENDMETHOD.
```

ONACTIONFORGET

Lanza la navegación a la vista que contiene el formulario mediante el conector de salida asociado.

```
method ONACTIONFORMULARO.
  wd_this->fire_out_formulari_plg().
endmethod.
```

ONACTIONOK

Comprobación que las claves introducidas por el usuario son correctas y entrada a la gestión de clientes y de pedidos

```
METHOD onactionok.

  DATA lo_nd_zusuario TYPE REF TO if_wd_context_node.
  DATA lo_el_zusuario TYPE REF TO if_wd_context_element.
  DATA ls_zusuario TYPE wd_this->element_zusuario.
  DATA lv_usuario LIKE ls_zusuario-usuario.
  DATA lv_password LIKE ls_zusuario-password.
  DATA wa_usuarios TYPE zusuarios.
  DATA lo_api_controller     TYPE REF TO if_wd_controller.
  DATA lo_message_manager    TYPE REF TO if_wd_message_manager.
  lo_nd_zusuario = wd_context->get_child_node( name = wd_this-
>wdctx_zusuario ).

* get element via lead selection
```

```
  lo_el_zusuario = lo_nd_zusuario->get_element(  ).
* get single attribute
  lo_el_zusuario->get_attribute(
    EXPORTING
      name =  `USUARIO`
    IMPORTING
      value = lv_usuario ).
* get single attribute
  lo_el_zusuario->get_attribute(
    EXPORTING
      name =  `PASSWORD`
    IMPORTING
      value = lv_password ).

  CALL FUNCTION 'CONVERSION_EXIT_ALPHA_INPUT'
    EXPORTING
      input  = lv_usuario
    IMPORTING
      output = lv_usuario.

  SELECT SINGLE * FROM zusuario INTO wa_usuarios
  WHERE usuario = lv_usuario AND   pass = lv_password.
  IF sy-subrc = 0.
       wd_this->fire_out_main_plg( ).
  ELSE.
      lo_api_controller ?= wd_this->wd_get_api( ).
      CALL METHOD lo_api_controller->get_message_manager
      RECEIVING
        message_manager = lo_message_manager.
*       report message
      CALL METHOD lo_message_manager->raise_error_message
      EXPORTING
        message_text = text-010.
    CALL METHOD lo_nd_zusuario->set_static_attributes_null.
  ENDIF.

ENDMETHOD.
```

7.3 El controlador formulario

7.3.1 Contexto

Las clases que se utilizan en el contexto formulario son las siguientes

Clase	Estructura
Cliente: los datos del cliente definidos en la especificación	Zstruc_alv_cliente: contiene los estados por los que pasa el cliente mediante iconos.
Usuario: los datos del usuario definidos en la especificación	
Documento los datos de la documentación definidos en la especificación	
Historial los datos del historial definidos en la especificación	

Tabla 9: clases utilizadas en el controlador formulario

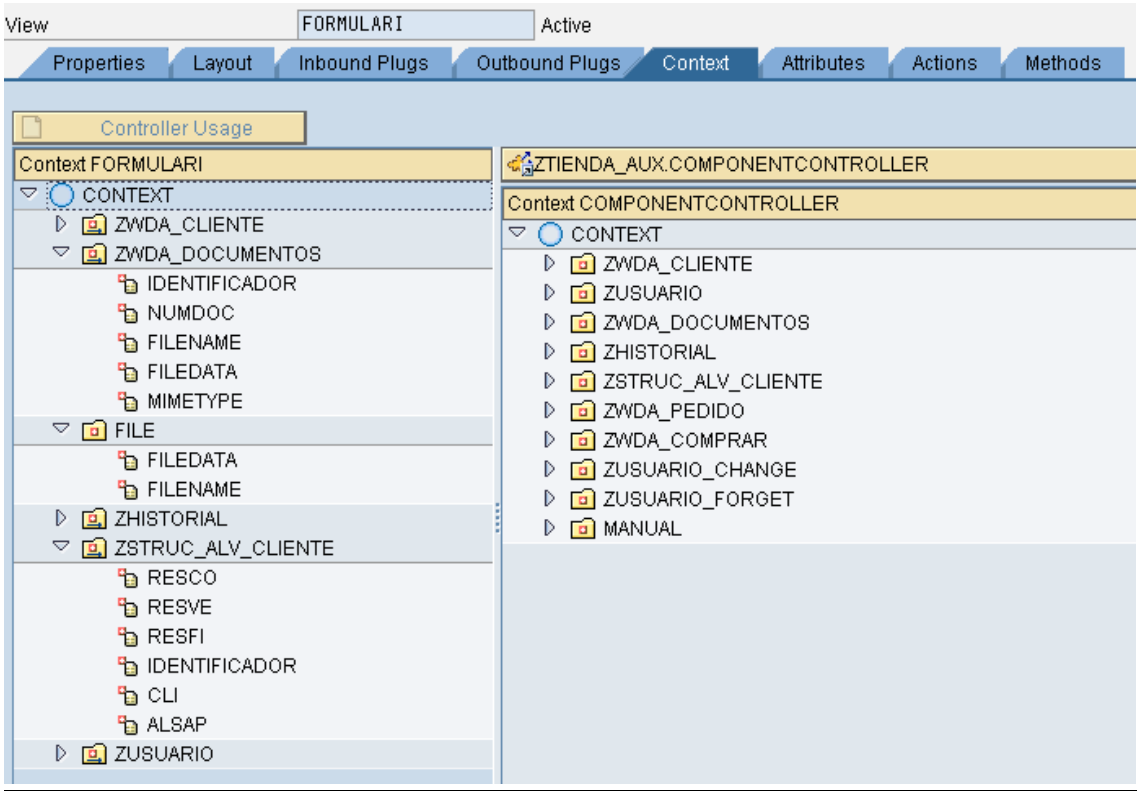


Figura 6: contexto del controlador formulario

7.3.2 Vista

En este apartado se muestra la vista creada para que el usuario rellene la información personal.

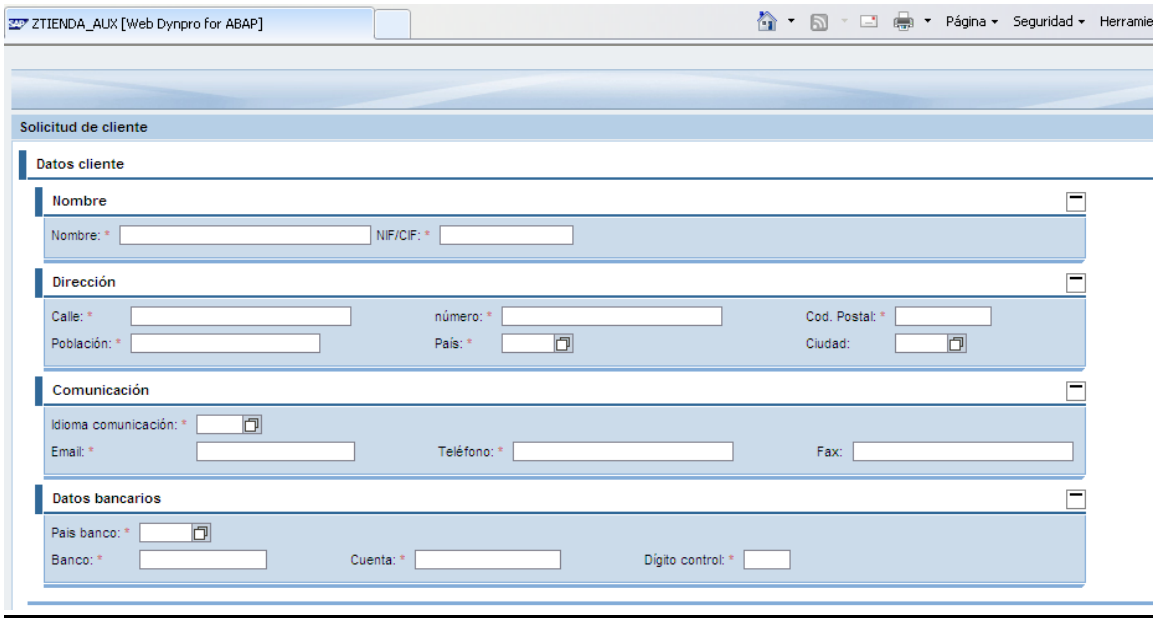


Figura 7: Interfaz de usuario para el formulario

La documentación aportada se anexará desde este apartado:

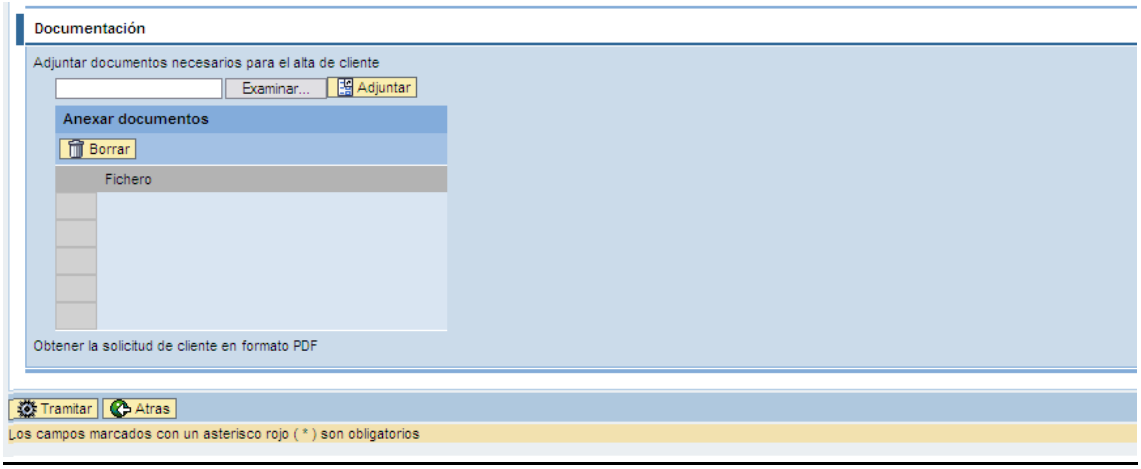


Figura 8: interfaz de usuario para el formulario

7.3.2.1 Conector entrada.

Al formulario se a accede desde la gestión de clientes en el momento de validar los datos por este motivo ha sido necesario crear 2 conectores de entrada. En función del conector de entrada se modificala vista para que muestre unos elementos de Interfaz u otros:

GO_TO_FORMULARI →navegamos a la vista formulario desde la vista principal

GO_TO_FORMULARI→navegamos a la vista formulario desde la gestión de clientes y activamos botón para rechazar datos.

7.3.2.1 Conector salida:

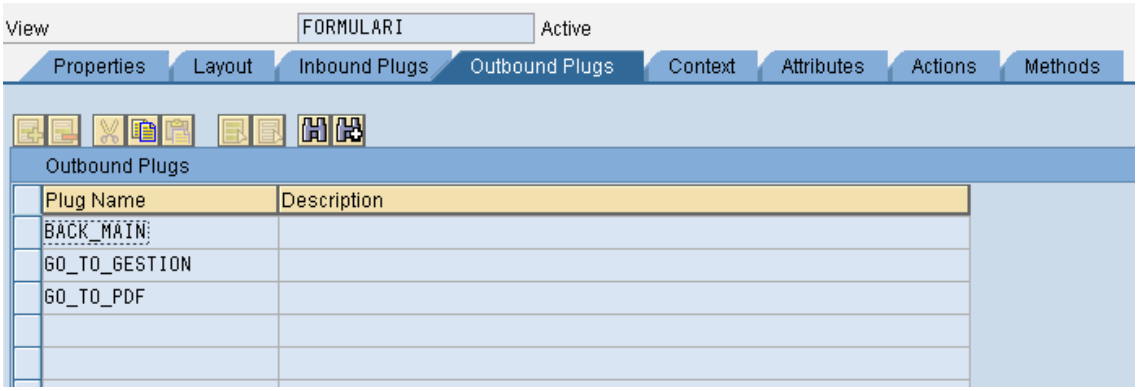


Figura 9: Conectores de entrada

GO_TO_GESTION →regresamos a la vista gestión de clientes

BACK_MAIN → regresamos a la vista autentificación.

7.3.3 Métodos

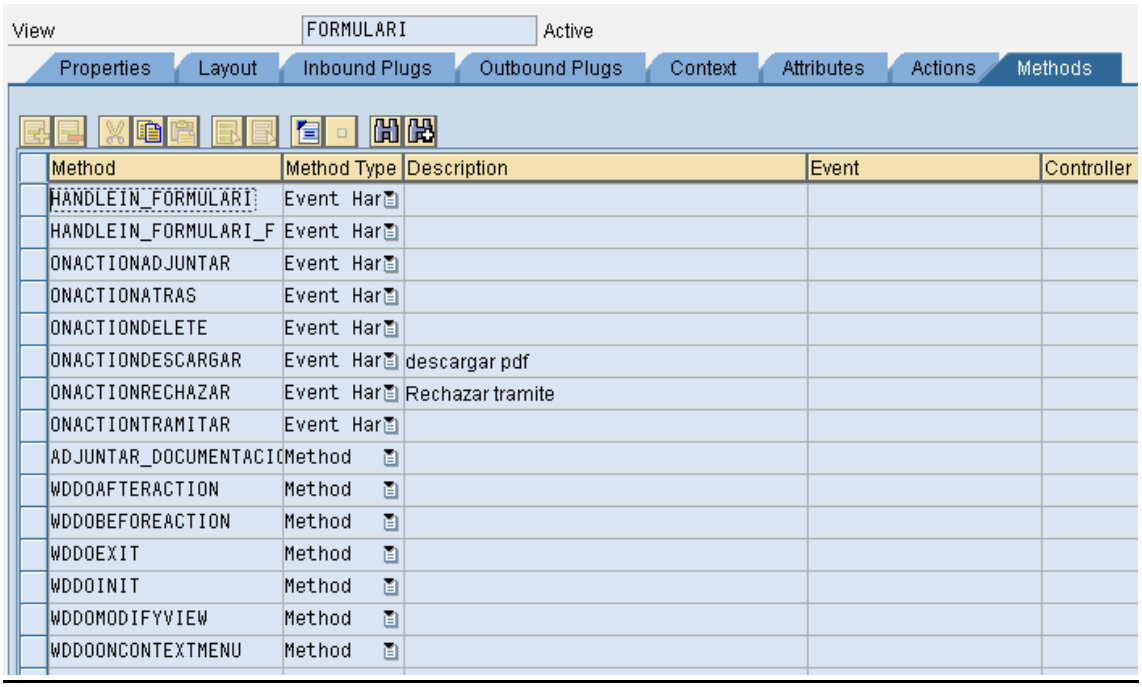


Figura 10: métodos del controlador formulario

HANDLEIN_FORMULARI

Este evento se dispara cuando se navega al formulario y se recupera la información y documentación del cliente en caso de que exista, es decir ya se haya registrado anteriormente.

```
METHOD handlein_formulari.

  DATA l_parameters TYPE wdr_event_parameter_list, lv_name TYPE string.
* leer identificador de gestion
  DATA lo_nd_zstruc_alv_cliente TYPE REF TO if_wd_context_node.
  DATA lo_el_zstruc_alv_cliente TYPE REF TO if_wd_context_element.
  DATA ls_zstruc_alv_cliente TYPE wd_this->element_zstruc_alv_cliente.
  DATA lv_identificador LIKE ls_zstruc_alv_cliente-identificador.
  DATA lo_nd_zwda_documentos TYPE REF TO if_wd_context_node.
  DATA lo_el_zwda_documentos TYPE REF TO if_wd_context_element.
  DATA ls_zwda_documentos TYPE wd_this->element_zwda_documentos.
  DATA lo_nd_zwda_cliente TYPE REF TO if_wd_context_node.
  DATA lo_el_zwda_cliente TYPE REF TO if_wd_context_element.
  DATA ls_zwda_cliente TYPE wd_this->element_zwda_cliente.
  DATA lit_documentos TYPE STANDARD TABLE OF zwda_documentos.

* navigate from <CONTEXT> to <ZSTRUC_ALV_CLIENTE> via lead selection
  lo_nd_zstruc_alv_cliente = wd_context->get_child_node( name = wd_this-
>wdctx_zstruc_alv_cliente ).
* @TODO handle not set lead selection
  IF NOT lo_nd_zstruc_alv_cliente IS INITIAL.
* get element via lead selection
```

```
    lo_el_zstruc_alv_cliente = lo_nd_zstruc_alv_cliente->get_element(  ).
* @TODO handle not set lead selection
    IF NOT lo_el_zstruc_alv_cliente IS INITIAL.
      lo_el_zstruc_alv_cliente->get_attribute(
        EXPORTING
          name =  `IDENTIFICADOR`
        IMPORTING
          value = lv_identificador ).

     SPLIT lv_identificador  AT space INTO lv_identificador lv_name.

      CALL FUNCTION 'CONVERSION_EXIT_ALPHA_INPUT'
        EXPORTING
          input  = lv_identificador
        IMPORTING
          output = lv_identificador.
* binding cliente
      lo_nd_zwda_cliente = wd_context->get_child_node( name = wd_this-
>wdctx_zwda_cliente ).
* get element via lead selection
      lo_el_zwda_cliente = lo_nd_zwda_cliente->get_element(  ).
      SELECT SINGLE * FROM zwda_cliente
      INTO ls_zwda_cliente WHERE identificador = lv_identificador.

      CALL FUNCTION 'CONVERSION_EXIT_ALPHA_OUTPUT'
        EXPORTING
          input  = lv_identificador
        IMPORTING
          output = ls_zwda_cliente-identificador.
      CALL METHOD lo_nd_zwda_cliente->set_static_attributes
         EXPORTING
           static_attributes = ls_zwda_cliente.
* obtener documentación asociada al formulario
      SELECT * FROM zwda_documentos INTO TABLE lit_documentos
      WHERE identificador = lv_identificador.

      lo_nd_zwda_documentos = wd_context->get_child_node( name = wd_this-
>wdctx_zwda_documentos ).
      IF NOT lo_nd_zwda_documentos IS INITIAL.
        CALL METHOD lo_nd_zwda_documentos->bind_table
          EXPORTING
            new_items = lit_documentos.
      ENDIF.
    ENDIF.
  ENDIF.
ENDMETHOD.
```

ONACTIONADJUNTAR

Esta acción se dispara cuando el usuario pulsa el botón para anexar documentación.

Los ficheros se guardan en tablas intermedias.

```
METHOD onactionadjuntar.

  DATA lo_nd_file TYPE REF TO if_wd_context_node.
  DATA lo_el_file TYPE REF TO if_wd_context_element.
  DATA ls_file TYPE wd_this->element_file.
  DATA : size        TYPE   i , l_fichero   TYPE  skwf_filnm,
         l_mimetype  TYPE  skwf_mime,l_name TYPE  sdba_actid,
         l_extension TYPE  sdba_funct, l_doc_type  TYPE  saedoktyp.
  DATA : elem_context   TYPE   REF   TO  if_wd_context_element,
         node_documents TYPE   REF   TO  if_wd_context_node,
         elem_documents TYPE   REF   TO  if_wd_context_element,
         l_numord       TYPE   numc10,
         stru_documents TYPE  wd_this->element_zwda_documentos,
```

```
         it_documents   TYPE   STANDARD   TABLE   OF  wd_this-
>element_zwda_documentos,
         wa_documents   TYPE  wd_this->element_zwda_documentos,
         e_logn_filename  TYPE  chkfile, lv_tabix  TYPE  sytabix.
DATA lo_nd_zwda_documentos TYPE REF TO if_wd_context_node.
  lo_nd_file = wd_context->get_child_node( name = wd_this->wdctx_file ).
  lo_el_file = lo_nd_file->get_element(  ).
* get all declared attributes
  lo_el_file->get_static_attributes(
    IMPORTING
      static_attributes = ls_file ).
* si es un tipo mime admitido por sap
  IF   NOT   ls_file-filedata IS INITIAL.
    MOVE   ls_file-filename  TO  l_fichero.
    CALL FUNCTION 'SKWF_MIMETYPE_OF_FILE_GET'
      EXPORTING
        filename = l_fichero
      IMPORTING
        mimetype = l_mimetype.
    SELECT  SINGLE  doc_type  INTO  l_doc_type
    FROM  toadd WHERE  mimetype  EQ  l_mimetype.
    IF  sy-subrc =  0 .
      e_logn_filename = ls_file-filename.
      CALL FUNCTION 'SPLIT_FILENAME'
        EXPORTING
          long_filename  = e_logn_filename
        IMPORTING
          pure_filename  = l_name
          pure_extension = l_extension.
      stru_documents-filedata = ls_file-filedata.
      stru_documents-mimetype = l_mimetype.
      CONCATENATE  l_name  '.' l_extension INTO  stru_documents-filename.
      lo_nd_zwda_documentos = wd_context->get_child_node( name = wd_this-
>wdctx_zwda_documentos ).
      IF NOT lo_nd_zwda_documentos IS INITIAL.
        DATA: it_zwda_documentos TYPE STANDARD TABLE OF zwda_document,
              wa_zwda_documentos TYPE zwda_documentos,
              num_lines TYPE i.
        CALL METHOD lo_nd_zwda_documentos->get_static_attributes_table
          IMPORTING
            table = it_zwda_documentos.
         describe table it_zwda_documentos lines num_lines.
         num_lines = num_lines + 1.
         stru_documents-numdoc = num_lines.
         APPEND stru_documents TO it_zwda_documentos.
         CALL METHOD lo_nd_zwda_documentos->bind_table
           EXPORTING
             new_items = it_zwda_documentos.
      ENDIF.
    ENDIF.
  ENDIF.
ENDMETHOD.
```

ONACTIONATRAS

Esta acción se dispara cuando el usuario pulsa el botón para volver a la vista principal mediante el conector de salida BACK_MAIN.

```
METHOD onactionatras .
  wd_this->fire_back_main_plg(  ).
ENDMETHOD.
```

ONACTIONDELETE

Esta acción se dispara cuando el usuario pulsa el botón para eliminar un documento anexado anteriormente. Se lee la tabla intermedia que contiene los ficheros y se elimina el documento.

```
METHOD onactiondelete .
  DATA lo_nd_zwda_documentos TYPE REF TO if_wd_context_node.
  DATA lo_el_zwda_documentos TYPE REF TO if_wd_context_element.
  DATA ls_zwda_documentos TYPE wd_this->element_zwda_documentos.
  DATA it_zwda_documentos TYPE STANDARD TABLE OF wd_this-
>element_zwda_documentos.
* navigate from <CONTEXT> to <ZWDA_DOCUMENTOS> via lead selection
  lo_nd_zwda_documentos = wd_context->get_child_node( name = wd_this-
>wdctx_zwda_documentos ).
* @TODO handle not set lead selection
  IF NOT lo_nd_zwda_documentos IS INITIAL.
* get element via lead selection
    lo_el_zwda_documentos = lo_nd_zwda_documentos->get_element(  ).
    IF NOT lo_el_zwda_documentos IS INITIAL.
* get all declared attributes
      lo_el_zwda_documentos->get_static_attributes(
        IMPORTING
          static_attributes = ls_zwda_documentos ).

      CALL METHOD lo_nd_zwda_documentos->get_static_attributes_table
        IMPORTING
          table = it_zwda_documentos.

      READ TABLE it_zwda_documentos
      WITH KEY numdoc = ls_zwda_documentos-numdoc TRANSPORTING NO FIELDS.
      IF sy-subrc = 0.
        DELETE it_zwda_documentos INDEX sy-tabix.
        CALL METHOD lo_nd_zwda_documentos->bind_table
          EXPORTING
            new_items            =  it_zwda_documentos
*           set_initial_elements = ABAP_TRUE
*           index                =
            .
      ENDIF.
    ENDIF.
  ENDIF.
ENDMETHOD.
```

ONACTIONRECHAZAR

Esta acción se dispara cuando el usuario pulsa el botón para rechazar la solicitud de cliente. Se hacen las siguientes comprobaciones:

- comprobamos que el rol asignado el usuario le permite ejecutar esta operación
- comprobamos que la solicitud de cliente no está ya dada de alta en SAP.

```
METHOD onactionrechazar.

  DATA lo_componentcontroller TYPE REF TO ig_componentcontroller .
  lo_componentcontroller =   wd_this->get_componentcontroller_ctr( ).
  DATA lo_api_controller     TYPE REF TO if_wd_controller.
  DATA lo_nd_zwda_cliente TYPE REF TO if_wd_context_node.
  DATA lo_el_zwda_cliente TYPE REF TO if_wd_context_element.
  DATA ls_zwda_cliente TYPE wd_this->element_zwda_cliente.
  DATA lv_circuito      TYPE char10, lv_tramite       TYPE char10.
```

```
  DATA lv_identificador TYPE char20, l_correcto       TYPE xfeld.
  DATA lo_message_manager    TYPE REF TO if_wd_message_manager.
  lo_api_controller ?= wd_this->wd_get_api( ).

  CALL METHOD lo_api_controller->get_message_manager
    RECEIVING
      message_manager = lo_message_manager.
  lo_nd_zwda_cliente = wd_context->get_child_node( name = wd_this-
>wdctx_zwda_cliente ).
* get element via lead selection
  lo_el_zwda_cliente = lo_nd_zwda_cliente->get_element(  ).
* get single attribute
  lo_el_zwda_cliente->get_attribute(
    EXPORTING
      name =  `CIRCUITO`
    IMPORTING
      value = lv_circuito ).
* get single attribute
  lo_el_zwda_cliente->get_attribute(
    EXPORTING
      name =  `TRAMITE`
    IMPORTING
      value = lv_tramite ).
* get single attribute
  lo_el_zwda_cliente->get_attribute(
    EXPORTING
      name =  `IDENTIFICADOR`
    IMPORTING
      value = lv_identificador ).
  CALL FUNCTION 'CONVERSION_EXIT_ALPHA_INPUT'
    EXPORTING
      input  = lv_identificador
    IMPORTING
      output = lv_identificador.

* comprobar rol-tramite
  lo_componentcontroller->validar_rol_tramite(
      EXPORTING
        identificador = lv_identificador
        tramite       = lv_tramite
      IMPORTING
        correcto      = l_correcto
    ).

  IF l_correcto = 'X'.
* comprobar que no está en tramite ALSAP
    DATA l_finalizado TYPE xfeld.

    lo_componentcontroller->finalizado_circuito(
      EXPORTING
        circuito =         lv_circuito
        identificador =   lv_identificador
      IMPORTING
        finalizado =      l_finalizado ).
    IF l_finalizado NE 'X'.

        lo_componentcontroller->retroceder_tramite(
        circuito =                          lv_circuito
        identificador =                     lv_identificador
        tramite =                           lv_tramite
      ).
* navegar a gestión.
        wd_this->fire_go_to_gestion_plg( ).
      ELSE.
*       report message
        CALL METHOD lo_message_manager->raise_error_message
          EXPORTING
```

```
            message_text = text-011.
      ENDIF.
    ELSE.
*         report message
      CALL METHOD lo_message_manager->raise_error_message
        EXPORTING
message_text = text 012'La solicitud de cliente ya ha completado el circuito'.
    ENDIF.
  ELSE.
    CALL METHOD lo_message_manager->raise_error_message
      EXPORTING
        message_text = text 013 'No tiene permiso para validar este trámite'.
  ENDIF.
ENDMETHOD.
```

ONACTIONTRAMITAR

Esta acción se dispara cuando el usuario pulsa el botón para validar la solicitud de cliente. Se hacen las siguientes comprobaciones:

- comprobación de que todos los campos obligatorios están informados
- comprobar que el cód. postal es correcto
- comprobar que la cuenta bancaria es correcta

En el caso de que sea un usuario nuevo se registra en el sistema y en otros casos se avanza un trámite.

```
METHOD onactiontramitar.

  DATA lo_componentcontroller TYPE REF TO ig_componentcontroller .
* get message manager
  DATA lo_api_controller     TYPE REF TO if_wd_controller.
  DATA lo_message_manager    TYPE REF TO if_wd_message_manager.
  DATA: wa_zhistorial        TYPE zhistorial.
  DATA  l_primtram           TYPE char10.
  DATA: l_circuito TYPE char10,
        l_tramite  TYPE char10.
* si es un cliente
  DATA lo_nd_zwda_cliente TYPE REF TO if_wd_context_node.
  DATA lo_el_zwda_cliente TYPE REF TO if_wd_context_element.
  DATA ls_zwda_cliente TYPE wd_this->element_zwda_cliente.
  DATA is_empty TYPE abap_bool.
  DATA wa_usuario TYPE zusuarios.
  DATA lpass_out TYPE wt_authval.
  DATA lpass_in  TYPE wt_authval.
  DATA l_messag TYPE string.
  DATA lo_nd_zwda_documentos TYPE REF TO if_wd_context_node.
  DATA lo_el_zwda_documentos TYPE REF TO if_wd_context_element.
  DATA ls_zwda_documentos TYPE wd_this->element_zwda_documentos.
  DATA l_view_controller TYPE REF TO if_wd_view_controller.

lo_api_controller ?= wd_this->wd_get_api( ).
  l_view_controller = wd_this->wd_get_api( ).
  cl_wd_dynamic_tool=>check_mandatory_attr_on_view( view_controller = l_view_c
ontroller ).

  CALL METHOD l_view_controller->get_message_manager
    RECEIVING
      message_manager = lo_message_manager.
```

```
  CLEAR is_empty.

* report message
  CALL METHOD lo_message_manager->is_empty
    RECEIVING
      empty = is_empty.

* Comprobar que los campos obligatorios estan informados
  IF is_empty = 'X'.
    lo_componentcontroller =   wd_this->get_componentcontroller_ctr( ).
* navigate from <CONTEXT> to <ZWDA_CLIENTE> via lead selection
    lo_nd_zwda_cliente = wd_context->get_child_node( name = wd_this-
>wdctx_zwda_cliente ).
* get element via lead selection
    lo_el_zwda_cliente = lo_nd_zwda_cliente->get_element(  ).
* get all declared attributes
    lo_el_zwda_cliente->get_static_attributes(
      IMPORTING
        static_attributes = ls_zwda_cliente ).

* COMPROBAR COD POSTAL
    CALL FUNCTION 'ADDR_POSTAL_CODE_CHECK'
      EXPORTING
        country                          = 'ES'
     postal_code_city                    = ls_zwda_cliente-codpostal

     EXCEPTIONS
       country_not_valid                 = 1
       region_not_valid                  = 2
       postal_code_city_not_valid        = 3
       postal_code_po_box_not_valid      = 4
       postal_code_company_not_valid     = 5
       po_box_missing                    = 6
       postal_code_po_box_missing        = 7
       postal_code_missing               = 8
       postal_code_pobox_comp_missing    = 9
       po_box_region_not_valid           = 10
       po_box_country_not_valid          = 11
       pobox_and_poboxnum_filled         = 12
       OTHERS                            = 13
              .
    IF sy-subrc <> 0.
      l_messag = 'Código postal incorrecto'.
      CALL METHOD lo_message_manager->raise_error_message
        EXPORTING
          message_text = l_messag.
    ELSE.
* comprobar el nif / cif
      CALL FUNCTION 'TAX_NUMBER_CHECK'
        EXPORTING
          country              = 'ES'
          tax_code_1               = ls_zwda_cliente-dni
       EXCEPTIONS
         not_valid              = 1
         different_fprcd        = 2
         OTHERS                 = 3.
      IF sy-subrc <> 0.
        l_messag = text 013 'El Nif / Cif es incorrecto'.
        CALL METHOD lo_message_manager->raise_error_message
          EXPORTING
            message_text = l_messag.
      ELSE.

* comprobar los datos bancarios
        CALL FUNCTION 'FI_CHECK_BANK_ACCOUNT_ES'
          EXPORTING
```

```
            i_bank_account     = ls_zwda_cliente-cuenta
            i_bank_control_key = ls_zwda_cliente-clavecontrol
            i_bank_country     = ls_zwda_cliente-clavepais
            i_bank_number      = ls_zwda_cliente-banco
          EXCEPTIONS
            not_valid          = 1
            OTHERS             = 2.
        IF sy-subrc <> 0.
          l_messag = 'La cuenta bancaria es incorrecta'.
          CALL METHOD lo_message_manager->raise_error_message
            EXPORTING
              message_text = l_messag.
        ELSE.

* registro sistema
          IF  ls_zwda_cliente-identificador IS INITIAL.
            DATA l_error TYPE sy-subrc.
            lo_componentcontroller->obtener_identificador(
                IMPORTING
                  error = l_error
                CHANGING
                  identificador =  ls_zwda_cliente-identificador
              ).
            IF l_error = 0.
              lo_componentcontroller->obtener_circuito_tramite(
              CHANGING
                circuito =  l_circuito
                tramite =   l_tramite
            ).
*       insertar el cliente
            ls_zwda_cliente-circuito       = l_circuito.
            ls_zwda_cliente-tramite        = l_tramite.
            INSERT into zwda_cliente values ls_zwda_cliente.
*       primer paso del historial
            IF sy-subrc = 0.
              lo_componentcontroller->avanza_sig_tramite(
                circuito      =  l_circuito
                identificador =  ls_zwda_cliente-identificador
                tramite       =  l_tramite
              ).
              wd_this->adjuntar_documentacion(
                identificador = ls_zwda_cliente-identificador
              ).
*         obtener password
              lpass_in = ls_zwda_cliente-identificador.
              CALL FUNCTION 'IDWT_CIS_PASSWORD_HASH'
                EXPORTING
                  l_password = lpass_in
                IMPORTING
                  value      = lpass_out.

              wa_usuario-usuario = ls_zwda_cliente-identificador.
              TRANSLATE lpass_out TO UPPER CASE.
              wa_usuario-pass    = lpass_out.
              wa_usuario-rol     = 'CLI'.
              INSERT into zusuarios values wa_usuario.

              IF sy-subrc = 0.
*           enviar mail
                CALL FUNCTION 'ZWDA_MAIL'
                  EXPORTING
                    i_usuario = wa_usuario-usuario
                    i_pass    = wa_usuario-pass
                    i_mail    = ls_zwda_cliente-mail.
*           mostrar mensaje

                CONCATENATE text-015
```

```
                  'Se ha registrado la solicitud de cliente en el sistema reci
birá en su correo sus claves de acceso:'
       wa_usuariousuario ' ' wa_usuariopass INTO l_messag SEPARATED BY space

                  CALL METHOD lo_message_manager->report_success
                    EXPORTING
                            message_text = l_messag.
                  CALL METHOD lo_nd_zwda_cliente->invalidate.
                  ENDIF.
               ENDIF.
             ENDIF.
       * siguientes pasos del circuito
        ELSE
           wd_this->siguientes_pasos().
    ENDIF.
ENDMETHOD.
METHOD siguientes_pasos

CALL FUNCTION 'CONVERSION_EXIT_ALPHA_INPUT'
       EXPORTING
          input  = ls_zwda_cliente-identificador
       IMPORTING
           output = lv_identificador.

* comprobar rol-tramite
lo_componentcontroller->validar_rol_tramite(
       EXPORTING
          identificador = lv_identificador
          tramite       = ls_zwda_cliente-tramite
       IMPORTING
          correcto = l_correcto ).

IF l_correcto = 'X'.
* comprobar que no está en tramite ALSAP
  lo_componentcontroller->finalizado_circuito(
        EXPORTING
          circuito =         ls_zwda_cliente-circuito
          identificador =   lv_identificador
        IMPORTING
          finalizado =      l_finalizado ).
  IF l_finalizado NE 'X'
* actualizar el cliente
    CALL FUNCTION 'CONVERSION_EXIT_ALPHA_INPUT'
        EXPORTING
           input  = ls_zwda_cliente-identificador
        IMPORTING
           output = ls_zwda_cliente-identificador

     MODIFY zwda_cliente FROM ls_zwda_cliente.
     l_tramite  = ls_zwda_cliente-tramite.
     l_circuito = ls_zwda_cliente-circuito.
     lo_componentcontroller->avanza_sig_tramite(
     circuito      =  l_circuito
     identificador =  ls_zwda_cliente-identificador
     tramite       =  l_tramite ).

    wd_this->adjuntar_documentacion(identificador = ls_zwda_cliente-
identificador).
   lo_nd_zwda_documentos = wd_context->get_child_node( name = wd_this-
>wdctx_zwda_documentos ).

    CALL METHOD wd_this->limpia_campos ( ).
* navegar a gestión.
    wd_this->fire_go_to_gestion_plg( ).
* la solicitud de cliente ya ha completado el circuito
  ELSE.
*  report message
```

```
     CALL METHOD lo_message_manager->raise_error_message
     EXPORTING
message_text = text0-20'La solicitud de cliente ya ha completado el circuito'.
  ENDIF.
 ELSE.
  CALL METHOD lo_message_manager->raise_error_message
  EXPORTING
    message_text = 'No tiene permiso para validar este trámite'.
 ENDIF.

ENDMETHOD.
```

LIMPIA_CAMPOS

Este método se invoca para eliminar todos los datos introducidos por el usuario.

```
METHOD limpia_campos
IF NOT  lo_nd_zwda_documentos IS INITIAL.
       IF NOT  lo_el_zwda_documentos IS INITIAL.
* get element via lead selection
         lo_el_zwda_documentos = lo_nd_zwda_documentos->get_element(  ).
         CALL METHOD lo_nd_zwda_documentos->set_static_attributes_null.
       ENDIF.
    ENDIF.
    IF NOT lo_nd_zwda_cliente IS INITIAL.
      IF NOT lo_el_zwda_cliente IS INITIAL.
        CALL METHOD lo_nd_zwda_cliente->set_static_attributes_null.
      ENDIF.
    ENDIF.
ENDMETHOD.
```

ADJUNTAR_DOCUMENTACION

Este método se invoca para registrar en las tablas la documentación anexada por el usuario en el momento que se registra la solicitud en el sistema

```
METHOD adjuntar_documentacion.

  DATA l_identificador TYPE char20.
* Documentación
  DATA lo_nd_zwda_documentos TYPE REF TO if_wd_context_node.
  DATA lo_el_zwda_documentos TYPE REF TO if_wd_context_element.
  DATA ls_zwda_documentos TYPE wd_this->element_zwda_documentos.
  DATA lit_documentos TYPE STANDARD TABLE OF wd_this->element_zwda_documentos.
* navigate from <CONTEXT> to <ZWDA_DOCUMENTOS> via lead selection
  lo_nd_zwda_documentos = wd_context->get_child_node( name = wd_this-
>wdctx_zwda_documentos ).
  CALL FUNCTION 'CONVERSION_EXIT_ALPHA_INPUT'
    EXPORTING
      input  = identificador
    IMPORTING
      output = l_identificador.

* @TODO handle not set lead selection
  IF NOT  lo_nd_zwda_documentos IS INITIAL.
    CALL METHOD lo_nd_zwda_documentos->get_static_attributes_table
      IMPORTING
```

```
        table = lit_documentos.
    DATA wa_documentos TYPE zwda_documentos.
    LOOP AT lit_documentos INTO wa_documentos.
      wa_documentos-identificador = l_identificador.
      MODIFY lit_documentos FROM wa_documentos TRANSPORTING identificador.
    ENDLOOP.
    DELETE FROM zwda_documentos WHERE identificador = l_identificador.
    MODIFY zwda_documentos FROM TABLE lit_documentos.
  ENDIF.
ENDMETHOD.
```

7.4 El controlador gestión clientes

7.4.1 Contexto

Clase	Estructura
Cliente: los datos del cliente definidos en la especificación	Zstruc_alv_cliente: contiene los estados por los que pasa el cliente mediante iconos.
Usuario: los datos del usuario definidos en la especificación	
Historial los datos del historial definidos en la especificación	

Tabla 10: clases utilizadasen el controlador gestión de clientes

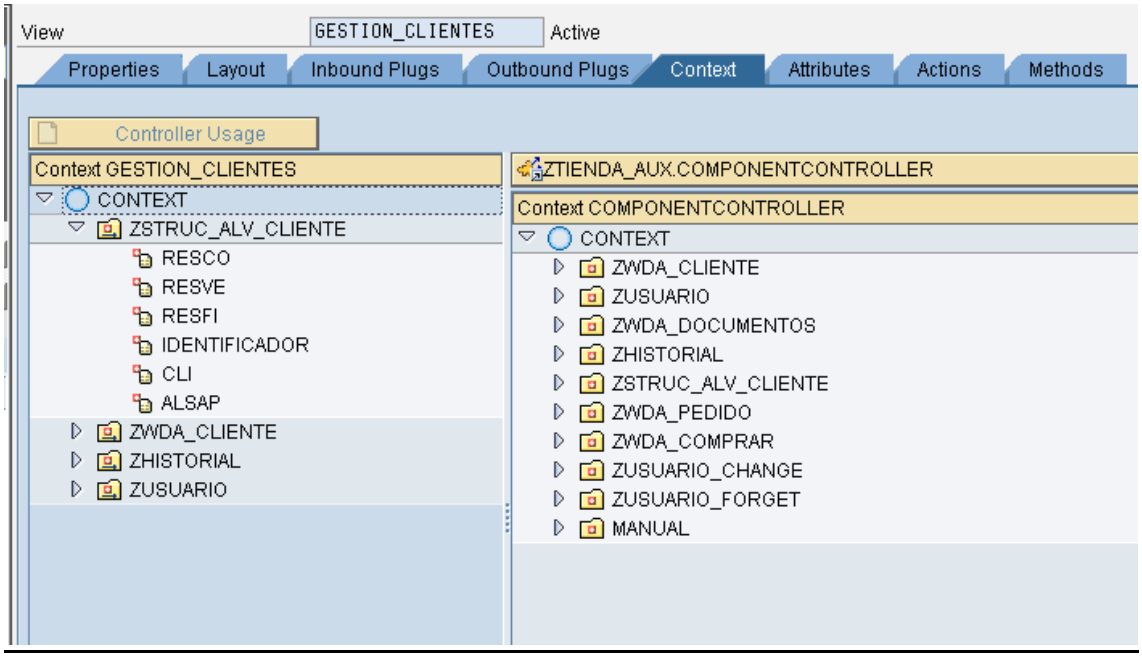


Figura 11: contexto en el controlador gestión de clientes

7.4.2 Vista

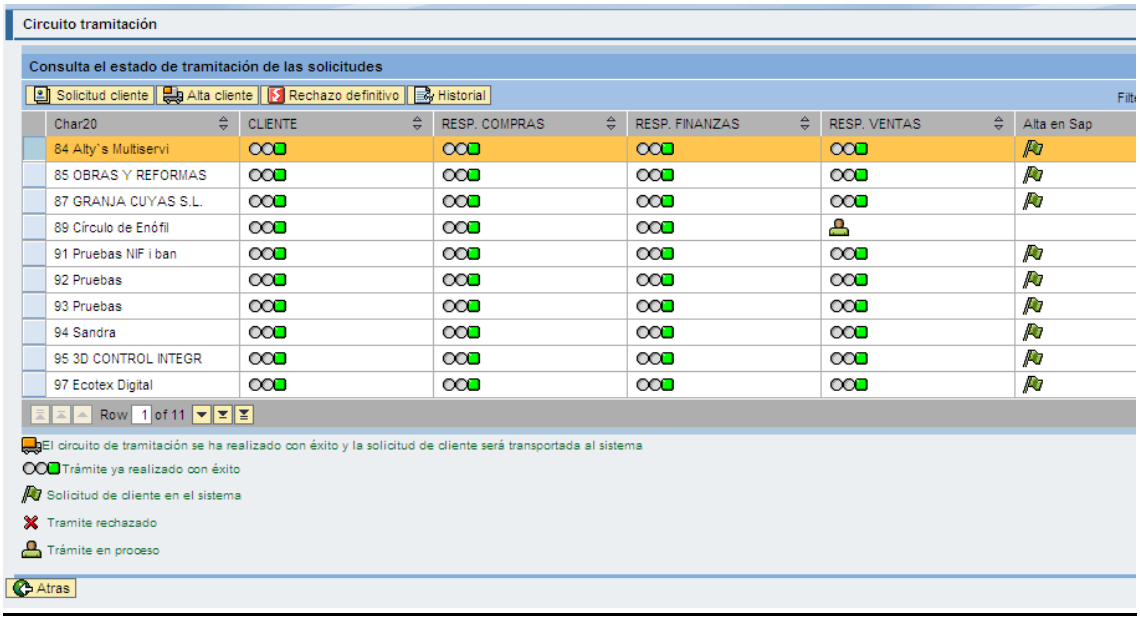


Figura 12: interfaz de usuario en controlador gestión de clientes

7.4.2.1 Conector entrada:

IN_GESTION_CLIENTES →navegamos a la vista gestión de clientes.

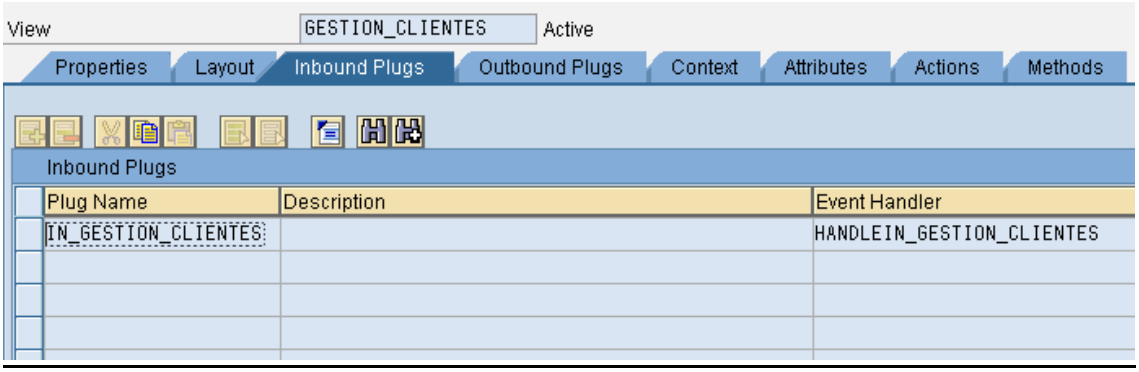


Figura 13: conectores de entrada

7.4.2.2 Conector salida:

BACK_TO_MAIN →navegamos a la vista principal.

GO_TO_HISTORIAL →navegamos al historial de la solicitud de cliente

GO_TO_FORMULARI →navegamos a la solicitud de cliente formulario

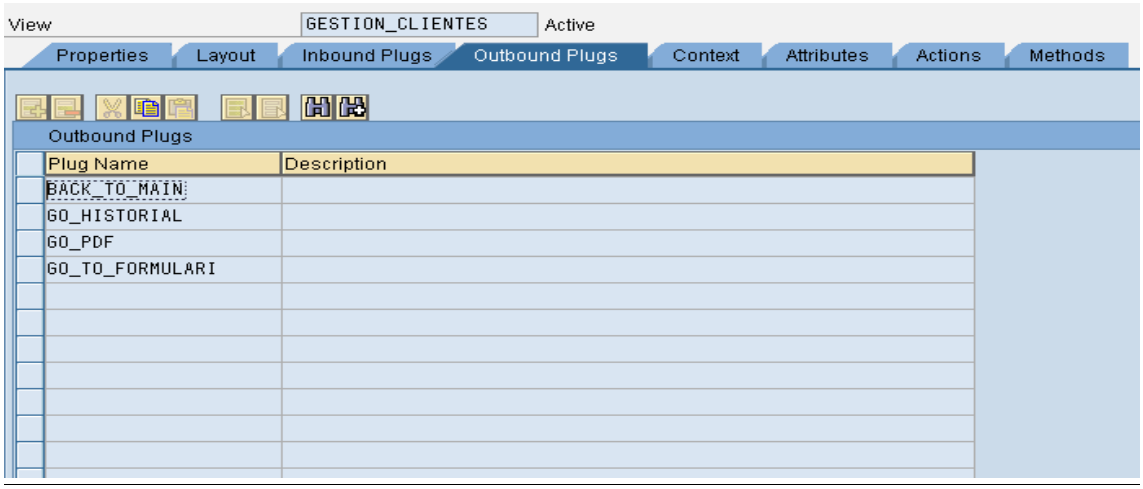


Figura 14: conectores de salida

7.4.3 Métodos

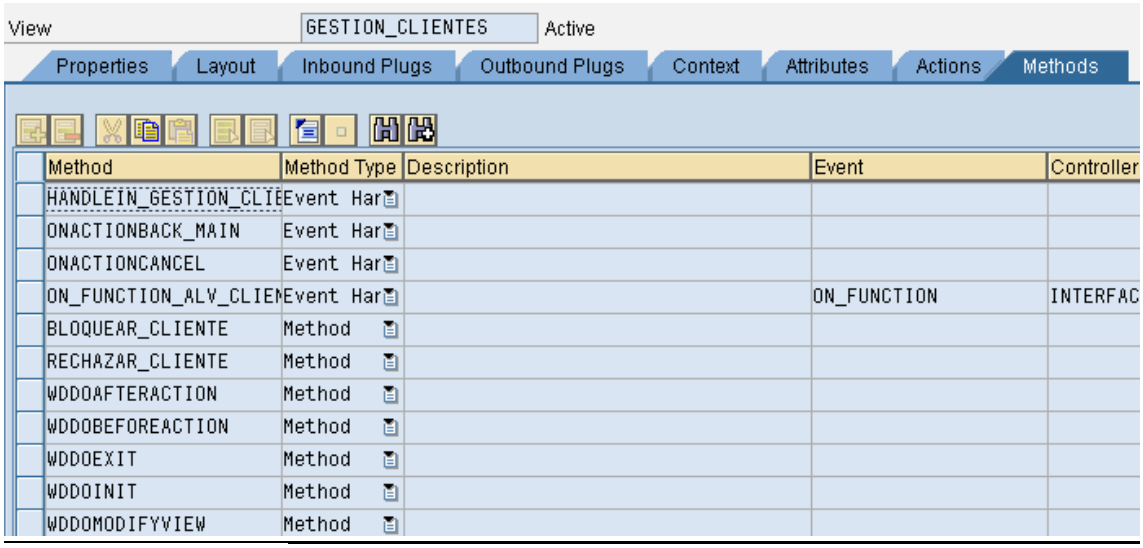


Figura 15: métodos

ADJUNTAR_DOCUMENTACION

Este método se invoca para registrar en las tablas la documentación anexada por el usuario en el momento que se registra la solicitud en el sistema

```
METHOD handlein_gestion_clientes .
  DATA lo_nd_zstruc_alv_cliente TYPE REF TO if_wd_context_node.
  DATA lo_el_zstruc_alv_cliente TYPE REF TO if_wd_context_element.
  DATA ls_zstruc_alv_cliente TYPE wd_this->element_zstruc_alv_cliente.
  DATA lit_gestion TYPE STANDARD TABLE OF wd_this->element_zstruc_alv_cliente.
  DATA lo_componentcontroller TYPE REF TO ig_componentcontroller .
  lo_componentcontroller =   wd_this->get_componentcontroller_ctr( ).
  DATA l_circuito TYPE char10.

* recuperem el circuit actiu i el primer tramit i l'ultim
  SELECT SINGLE circuito INTO l_circuito
         FROM  zcircuito WHERE active = 'X'.
```

```
  lo_componentcontroller->rellenar_clientes_alv(
    CHANGING
       circuito    = l_circuito
       it_gestion = lit_gestion ).

* refescar datos alv
  lo_nd_zstruc_alv_cliente = wd_context->get_child_node( name = wd_this-
>wdctx_zstruc_alv_cliente ).

* @TODO handle not set lead selection
  IF NOT lo_nd_zstruc_alv_cliente IS INITIAL.

    CALL METHOD lo_nd_zstruc_alv_cliente->bind_table
      EXPORTING
        new_items            = lit_gestion
*        set_initial_elements = ABAP_TRUE
*        index
  ENDIF.
ENDMETHOD.

METHOD onactionback_main .
  wd_this->fire_back_to_main_plg( ).
ENDMETHOD.
```

ON_FUNCTION_ALV_CLIENTES

Este evento se cuando el usuario ejecuta alguna de las funciones en el listado de clientes. Las acciones que puede ejecutar son las siguientes:

- Navegar solicitud de cliente (formulario)
- Generar cliente en SAP
- Visualizar historial
- Rechazar definitivamente la solicitud de cliente.

```
METHOD on_function_alv_clientes.

  DATA lo_nd_zstruc_alv_cliente TYPE REF TO if_wd_context_node.
  DATA lo_el_zstruc_alv_cliente TYPE REF TO if_wd_context_element.
  DATA ls_zstruc_alv_cliente TYPE wd_this->element_zstruc_alv_cliente.
  DATA lv_identificador LIKE ls_zstruc_alv_cliente-identificador.
  DATA: wa_zhistorial TYPE zhistorial.
  DATA lv_ident TYPE char20, lv_nombre TYPE string.
  DATA lo_componentcontroller TYPE REF TO ig_componentcontroller .

  lo_componentcontroller =   wd_this->get_componentcontroller_ctr( ).
  lo_nd_zstruc_alv_cliente = wd_context->get_child_node( name = wd_this-
>wdctx_zstruc_alv_cliente ).
* @TODO handle not set lead selection
  IF NOT lo_nd_zstruc_alv_cliente IS INITIAL.
* get element via lead selection
    lo_el_zstruc_alv_cliente = lo_nd_zstruc_alv_cliente->get_element(  ).
* @TODO handle not set lead selection
    IF NOT lo_el_zstruc_alv_cliente IS INITIAL.
* get single attribute
      lo_el_zstruc_alv_cliente->get_attribute(
        EXPORTING
          name =  `IDENTIFICADOR`
        IMPORTING
          value = lv_identificador ).
      SPLIT lv_identificador AT space INTO lv_ident lv_nombre.
```

```
      CALL FUNCTION 'CONVERSION_EXIT_ALPHA_INPUT'
        EXPORTING
          input  = lv_ident
        IMPORTING
          output = lv_ident.
    ENDIF.
  ENDIF.
  CASE  r_param->id.
* Navegar a la solicitud de cliente (formulario)
    WHEN 'FORMULARI'.
      DATA lo_nd_zusuario TYPE REF TO if_wd_context_node.
      DATA lo_el_zusuario TYPE REF TO if_wd_context_element.
      DATA ls_zusuario TYPE wd_this->element_zusuario.
      DATA lv_usuario LIKE ls_zusuario-usuario.
*       navigate from <CONTEXT> to <ZUSUARIO> via lead selection
      lo_nd_zusuario = wd_context->get_child_node( name = wd_this-
>wdctx_zusuario ).
      lo_el_zusuario = lo_nd_zusuario->get_element(  ).
      lo_el_zusuario->get_attribute(
        EXPORTING
          name =  `USUARIO`
        IMPORTING
          value = lv_usuario ).
      wd_this->fire_go_to_formulari_plg(
        identificador =  lv_ident
        usuario       =  lv_usuario ).

* dar de alta el cliente en Sap
    WHEN 'ALTA_SAP'.
      lo_componentcontroller->generar_cliente(
        identificador =  lv_ident ).
* Visualizar historial
    WHEN  'HISTORIAL'.
      wd_this->fire_go_historial_plg( ).

* Rechazar la solicitud del cliente
    WHEN  'RECHAZO'.
      wd_this->rechazar_cliente(
       identificador = lv_ident).
  ENDCASE.
ENDMETHOD.
```

RECHAZAR_CLIENTE

Se elimina definitivamente una solicitud de cliente en el sistema.

```
METHOD rechazar_cliente.
 DATA: l_ident TYPE char20.
  DATA: l_correcto TYPE xfeld, lit_gestion TYPE ztab_zstructalv.
  DATA: wa_cliente TYPE zwda_cliente, wa_bloqueo TYPE zbloqueos.
  DATA lo_nd_zstruc_alv_cliente TYPE REF TO if_wd_context_node.
  DATA lo_componentcontroller TYPE REF TO ig_componentcontroller .
  lo_componentcontroller =   wd_this->get_componentcontroller_ctr( ).
  DATA lo_api_controller     TYPE REF TO if_wd_controller.
  DATA lo_message_manager    TYPE REF TO if_wd_message_manager.
* get message manager
  lo_api_controller ?= wd_this->wd_get_api( ).
  CALL METHOD lo_api_controller->get_message_manager
    RECEIVING
      message_manager = lo_message_manager.
* comprobar permisos para rechazar
  lo_componentcontroller->validar_rol_tramite(
  EXPORTING
    identificador = identificador
    tramite       = 'RECHA'
```

```
  IMPORTING
    correcto      = l_correcto ).
  IF l_correcto = 'X'.
    CALL FUNCTION 'CONVERSION_EXIT_ALPHA_INPUT'
      EXPORTING
        input  = identificador
      IMPORTING
       output = l_ident.
    SELECT SINGLE * FROM zwda_cliente INTO wa_cliente
    WHERE identificador = l_ident.
* comprobar que no estar en trámite altasap
    DATA l_finalizado TYPE xfeld.
    lo_componentcontroller->finalizado_circuito(
      EXPORTING
        circuito =   wa_cliente-circuito                    " char10
        identificador = l_ident                             " char20
      IMPORTING
        finalizado =   l_finalizado                    ).
    IF l_finalizado NE 'X'.
      wa_cliente-rechazo = 'X'.
      MODIFY zwda_cliente FROM wa_cliente.
      lo_componentcontroller =   wd_this->get_componentcontroller_ctr( ).
      lo_componentcontroller->rellenar_clientes_alv(
         CHANGING
           circuito =    wa_cliente-circuito                " char10
           it_gestion =  lit_gestion                 ).
 lo_nd_zstruc_alv_cliente = wd_context->get_child_node( name = wd_this-
>wdctx_zstruc_alv_cliente ).
*       @TODO handle not set lead selection
      IF NOT lo_nd_zstruc_alv_cliente IS INITIAL.
        CALL METHOD lo_nd_zstruc_alv_cliente->bind_table
          EXPORTING
            new_items = lit_gestion.
      ENDIF.
    ELSE.
*       report message
      CALL METHOD lo_message_manager->raise_error_message
        EXPORTING
          message_text = 'La solicitud de cliente ya es apta para el alta, no
es posible rechazar'.
    ENDIF.
  ELSE.
*       report message
    CALL METHOD lo_message_manager->raise_error_message
      EXPORTING
        message_text = 'No tiene permiso para rechazar solicitud'.
  ENDIF.
ENDMETHOD.
```

7.5 El controlador historial

7.5.1 Contexto

Clase	Estructura
Historial los datos del historial definidos en la especificación	Zstruc_alv_cliente: contiene los estados por los que pasa el cliente mediante iconos.

Tabla 11: clases utilizadas

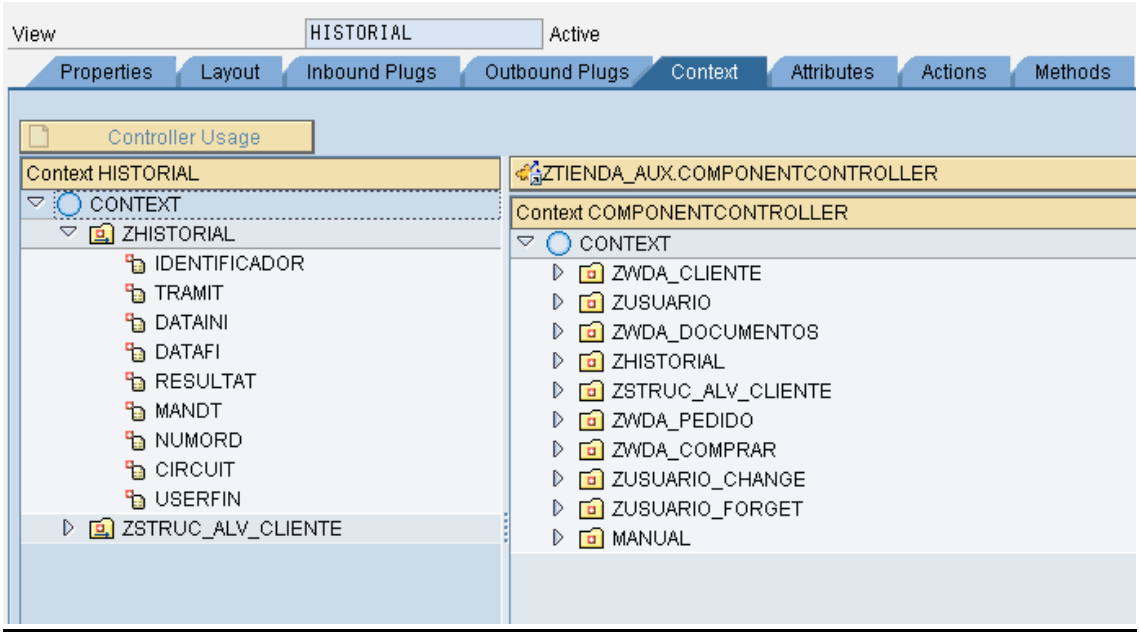


Figura 16: contexto del controlador historial

7.5.2 Vista

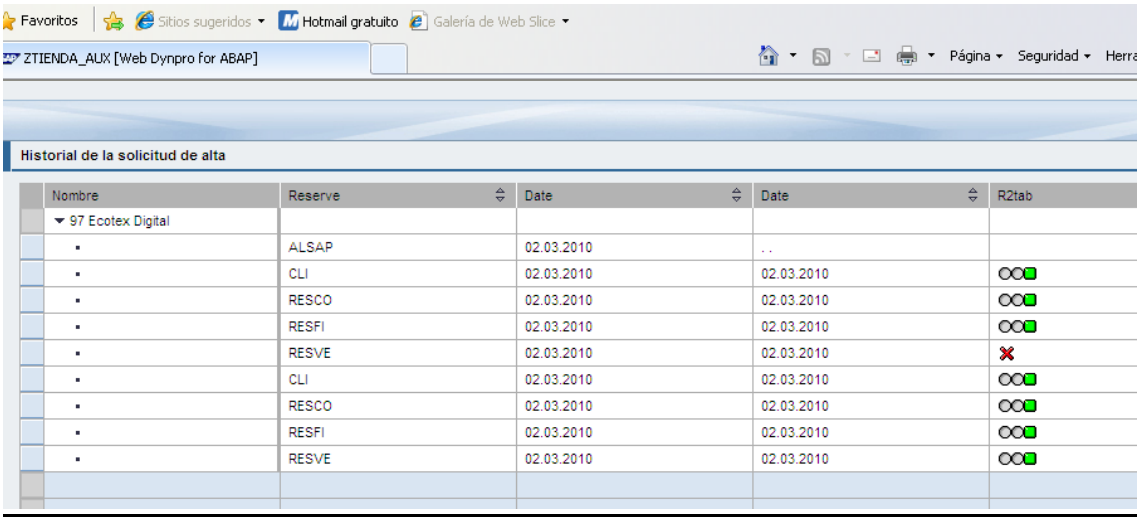


Figura 17: interfaz de usuario en el controlador historial

7.4.2.1 Conector entrada:

IN_HISTORIAL→navegamos a la vista que muestra el historial.

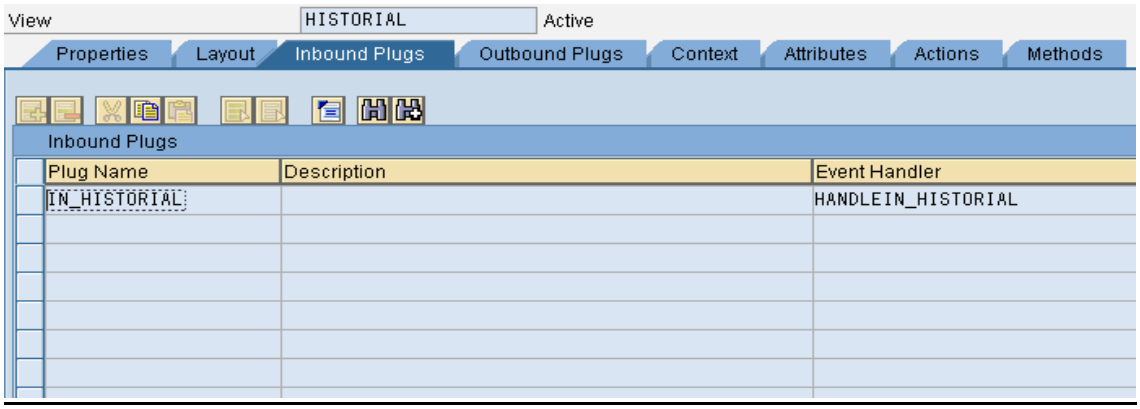


Figura 18: conector de entrada

7.4.2.1 Conector salida:

OUT_HISTORIAL→regresamos a gestión de clientes.

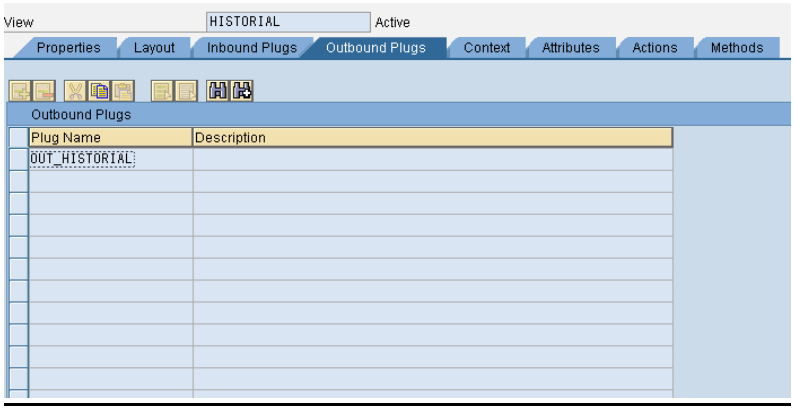


Figura 19: conector salida

7.5.3 Métodos

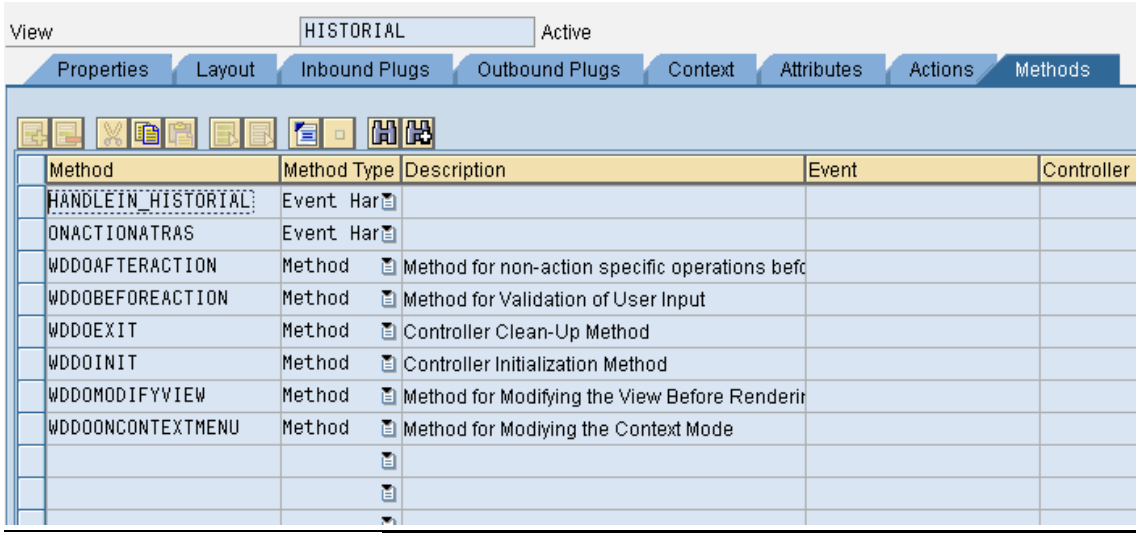


Figura 20: métodos

HANDLEIN_HISTORIAL: recuperamos toda la información asociada a una solicitud de cliente para mostrarla

```
METHOD handlein_historial.

  DATA it_historial TYPE STANDARD TABLE OF zhistorial.
  data wa_historial type zhistorial.
  DATA lo_nd_zstruc_alv_cliente TYPE REF TO if_wd_context_node.
  DATA lo_el_zstruc_alv_cliente TYPE REF TO if_wd_context_element.
  DATA ls_zstruc_alv_cliente TYPE wd_this->element_zstruc_alv_cliente.
  DATA lv_identificador LIKE ls_zstruc_alv_cliente-identificador.
*   navigate from <CONTEXT> to <ZSTRUC_ALV_CLIENTE> via lead selection
  lo_nd_zstruc_alv_cliente = wd_context->get_child_node( name = wd_this-
>wdctx_zstruc_alv_cliente ).
*   @TODO handle not set lead selection
  IF NOT lo_nd_zstruc_alv_cliente IS INITIAL.
*   get element via lead selection
    lo_el_zstruc_alv_cliente = lo_nd_zstruc_alv_cliente->get_element(  ).

*   @TODO handle not set lead selection
    IF NOT lo_el_zstruc_alv_cliente IS INITIAL.
*   get single attribute
      lo_el_zstruc_alv_cliente->get_attribute(
        EXPORTING
          name =  `IDENTIFICADOR`
        IMPORTING
          value = lv_identificador ).
      DATA lv_name TYPE string.

      SPLIT  lv_identificador AT space INTO lv_identificador lv_name.
      CALL FUNCTION 'CONVERSION_EXIT_ALPHA_INPUT'
        EXPORTING
          input  = lv_identificador
        IMPORTING
          output = lv_identificador.

      SELECT * FROM zhistorial
        INTO CORRESPONDING FIELDS OF TABLE it_historial
        WHERE identificador = lv_identificador

      LOOP AT it_historial INTO wa_historial.
        IF wa_historial-RESULTAT = 'OK'.
          wa_historial-RESULTAT = '@08@'.
        ELSEIF wa_historial-RESULTAT = 'KO'.
          wa_historial-RESULTAT = '@02@'.
        ENDIF.
        CALL FUNCTION 'CONVERSION_EXIT_ALPHA_OUTPUT'
          EXPORTING
            input  = wa_historial-identificador
          IMPORTING
            output = wa_historial-identificador.

        CONCATENATE  wa_historial-identificador lv_name
        INTO wa_historial-identificador
        SEPARATED BY space.
        MODIFY it_historial FROM wa_historial INDEX sy-tabix.
      ENDLOOP.

      DATA lo_nd_zhistorial TYPE REF TO if_wd_context_node.
      DATA lo_el_zhistorial TYPE REF TO if_wd_context_element.
      DATA ls_zhistorial TYPE wd_this->element_zhistorial.
*       navigate from <CONTEXT> to <ZHISTORIAL> via lead selection
      lo_nd_zhistorial = wd_context->get_child_node( name = wd_this-
>wdctx_zhistorial ).
```

```
*          @TODO handle not set lead selection
      IF NOT lo_nd_zhistorial IS INITIAL.

        CALL METHOD lo_nd_zhistorial->bind_table
          EXPORTING
            new_items = it_historial.

      ENDIF.
    ENDIF.
  ENDIF.
ENDMETHOD.
```

ONACTION_ATRAS

Acción asociada al botón que dispara el conector de salida.

```
method ONACTIONATRAS .
  wd_this->fire_out_historial_plg(
  ).
endmethod.
```

7.6 El controlador gestión pedidos

7.6.1 Contexto

Clase	Estructura
Cliente: los datos del cliente definidos en la especificación	Zwda_comprar: línea de pedido
Usuario: los datos del usuario definidos en la especificación	Zstruc_alv_cliente: contiene los estados por los que pasa el cliente mediante iconos.
Pedido: cabecera y líneas de pedido	

Tabla 12: clases utilizadas

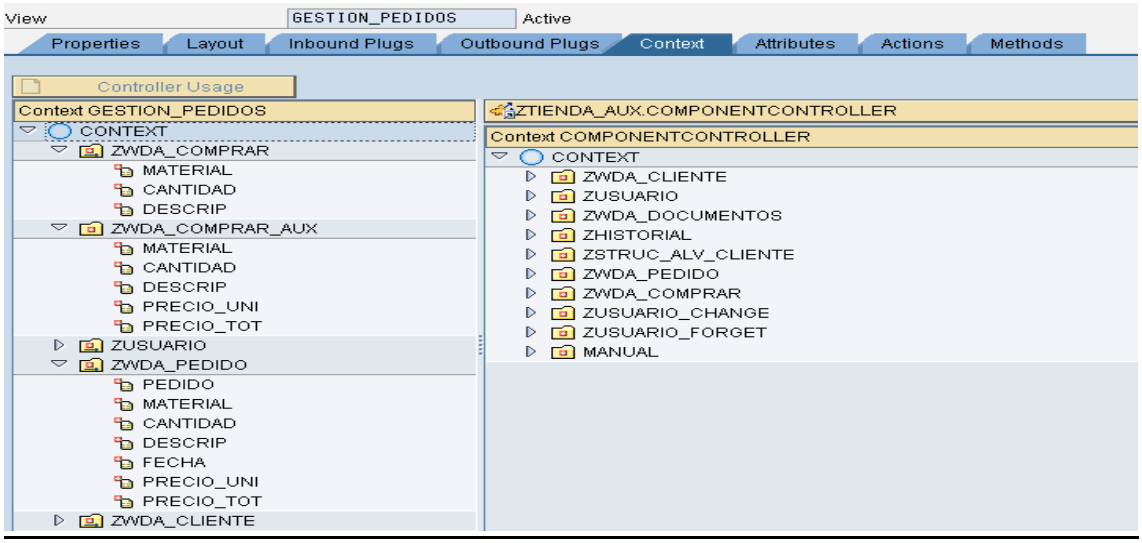


Figura 21: contexto controlador gestión de pedidos

7.6.2 Vista

7.6.2.1 Consultar pedidos

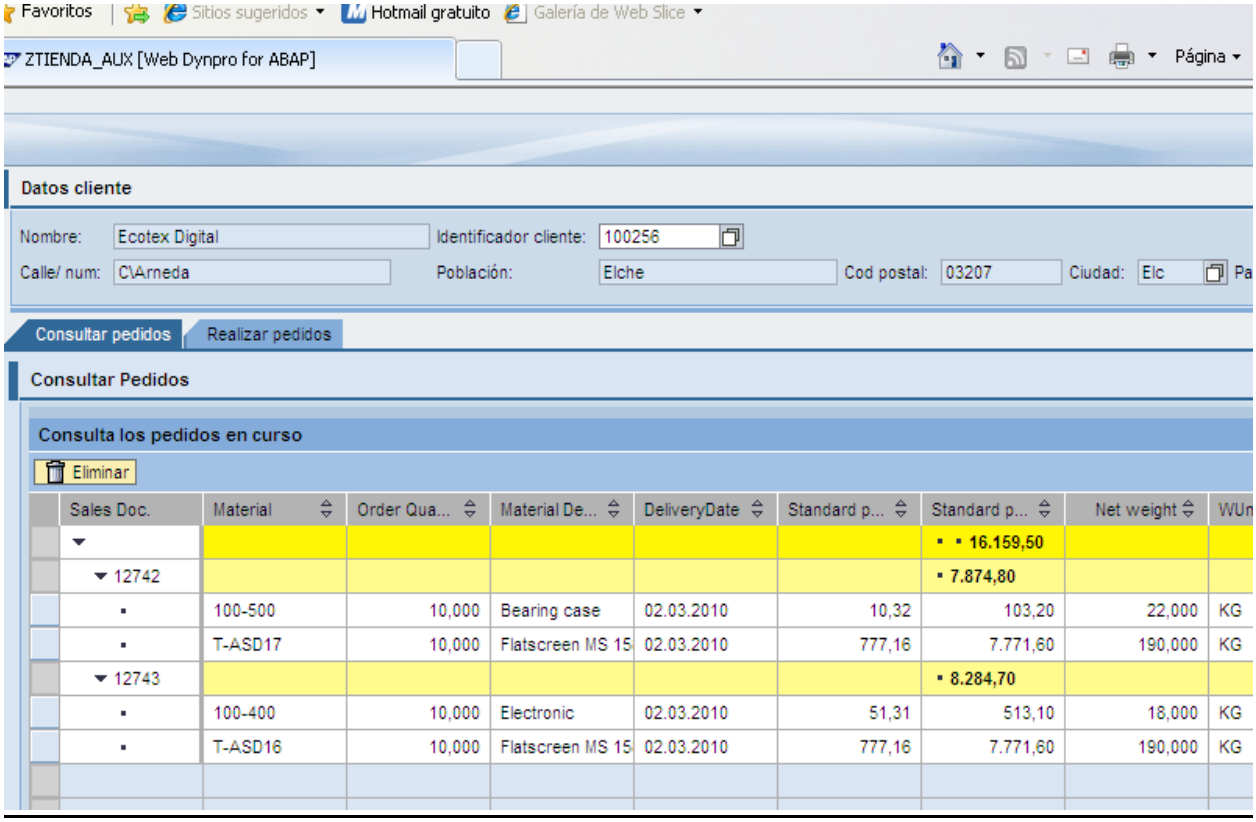


Figura 22: interfaz de usuario para consulta pedidos

7.6.2.2 Realizar pedidos

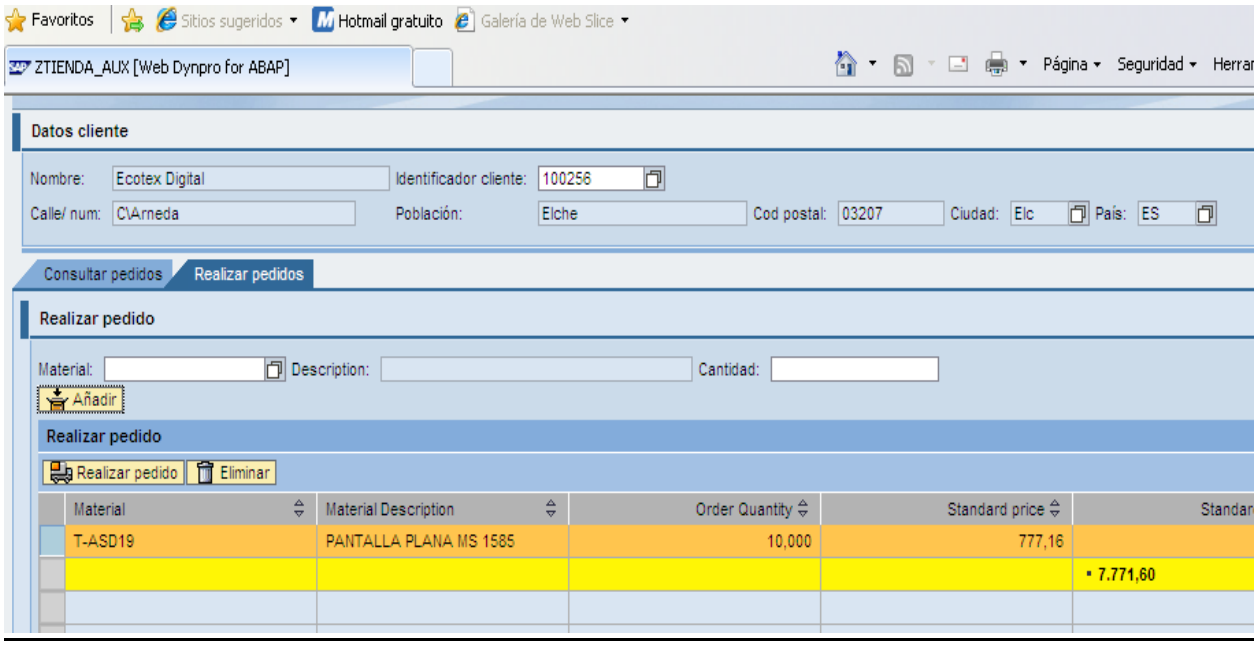


Figura 23: interfaz de usuario para realizar pedidos

7.6.2.3 Conectores entrada

IN_GESTION_PEDIDOS→entrada a la vista gestión de pedidos.

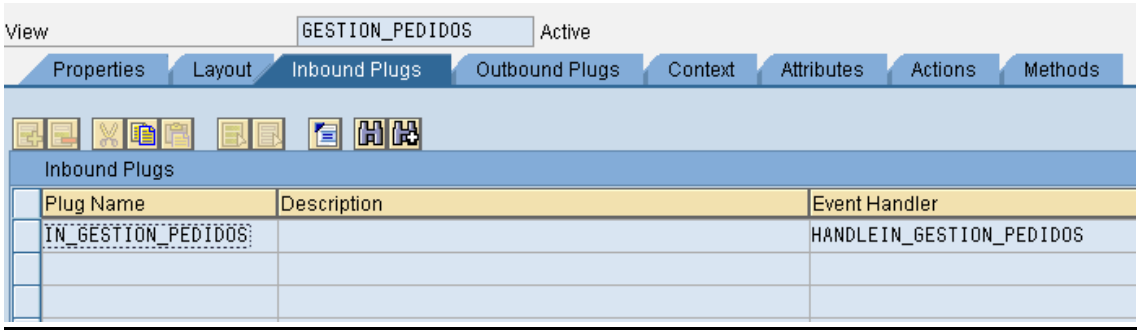


Figura 24: conector entrada

7.5.2.4 Conectores salida

BACK_MAIN→regreso a la vista principal.

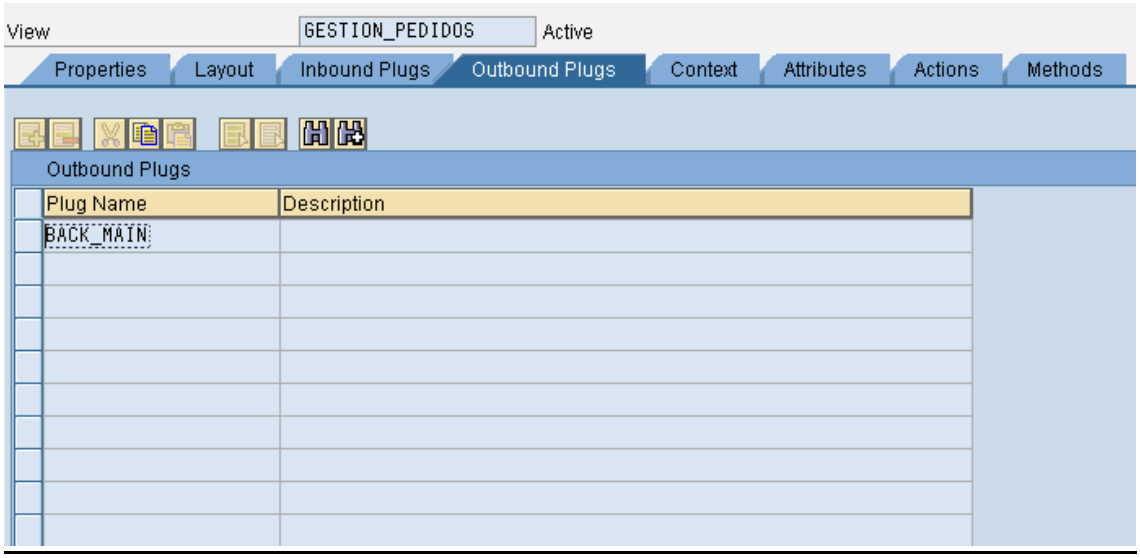


Figura 25: conector salida

7.6.3 Métodos

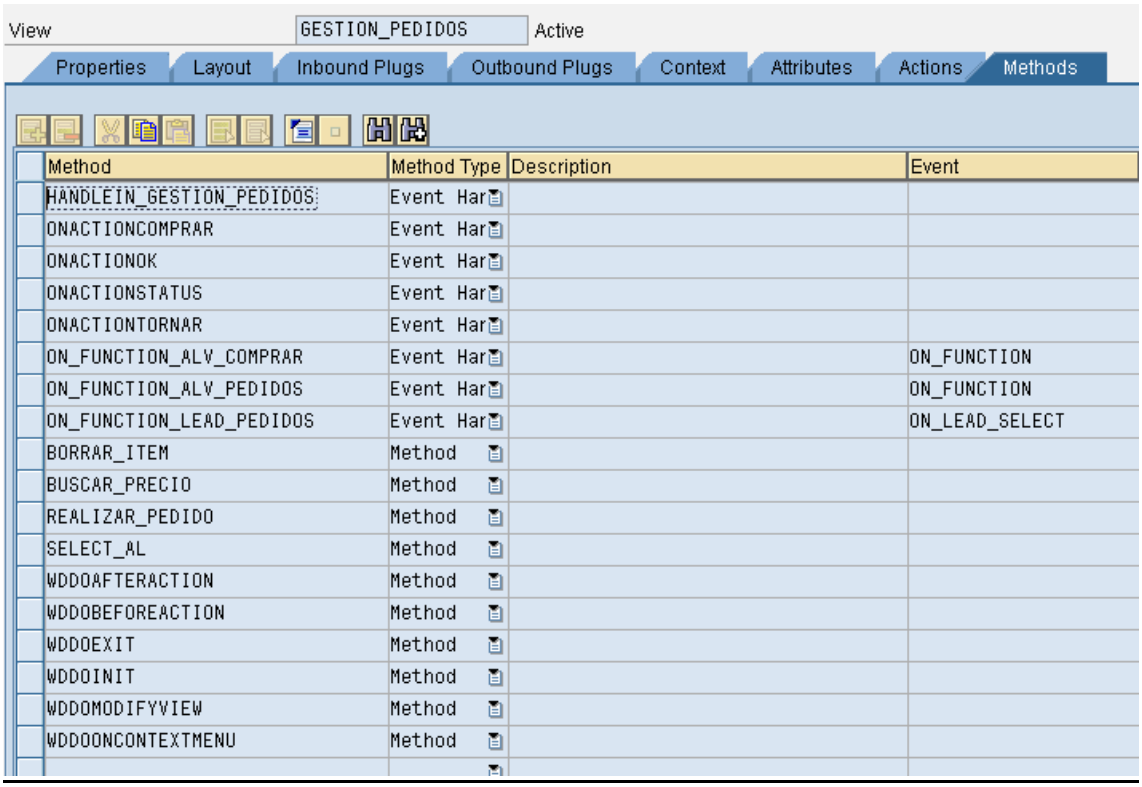


Figura 26: métodos

HANDLE_GESTION_PEDIDOS

Este método recupera de SAP todos los datos necesarios del cliente para poder realizar el pedido. Además obtiene todos los pedidos que tiene el cliente pendiente de entrega.

```
METHOD handlein_gestion_pedidos.

* 1 -> obtener cliente
  DATA lo_nd_zusuario TYPE REF TO if_wd_context_node.
  DATA lo_el_zusuario TYPE REF TO if_wd_context_element.
  DATA ls_zusuario TYPE wd_this->element_zusuario.
  DATA lv_usuario LIKE ls_zusuario-usuario.
  data wa_cliente type kna1.
  DATA lo_nd_zwda_cliente TYPE REF TO if_wd_context_node.
  DATA lo_el_zwda_cliente TYPE REF TO if_wd_context_element.
  DATA ls_zwda_cliente TYPE wd_this->element_zwda_cliente.
*   navigate from <CONTEXT> to <ZUSUARIO> via lead selection
  lo_nd_zusuario = wd_context->get_child_node( name = wd_this-
>wdctx_zusuario ).
*   get element via lead selection
  lo_el_zusuario = lo_nd_zusuario->get_element(  ).
*   get single attribute
  lo_el_zusuario->get_attribute(
    EXPORTING
```

```
      name =  `USUARIO`
    IMPORTING
      value = lv_usuario ).
  CALL FUNCTION 'CONVERSION_EXIT_ALPHA_INPUT'
    EXPORTING
      input  = lv_usuario
    IMPORTING
      output = lv_usuario.

  DATA wa_usuario TYPE zusuarios.
  SELECT SINGLE * INTO wa_usuario
  FROM zusuarios WHERE usuario =  lv_usuario.

* binding cliente
  SELECT SINGLE * FROM kna1 INTO wa_cliente
  WHERE kunnr = wa_usuario-cliente.

lo_nd_zwda_cliente = wd_context->get_child_node( name = wd_this-
>wdctx_zwda_cliente ).
lo_el_zwda_cliente = lo_nd_zwda_cliente->get_element(  ).

      CALL METHOD lo_el_zwda_cliente->set_attribute
        EXPORTING
          value = wa_cliente-name1
          name  = `NOMBRE`.
      CALL METHOD lo_el_zusuario->set_attribute
        EXPORTING
          value = wa_usuario-cliente
          name  = `CLIENTE`.
      CALL METHOD lo_el_zwda_cliente->set_attribute
        EXPORTING
          value = wa_cliente-stras
          name  = `CALLE`.
      CALL METHOD lo_el_zwda_cliente->set_attribute
        EXPORTING
          value = wa_cliente-ort01
          name  = `POBLACION`.

      CALL METHOD lo_el_zwda_cliente->set_attribute
        EXPORTING
          value = wa_cliente-telf1
          name  = `TELEFONO`.
      CALL METHOD lo_el_zwda_cliente->set_attribute
        EXPORTING
          value = wa_cliente-ort01
          name  = `CIUDAD`.
      CALL METHOD lo_el_zwda_cliente->set_attribute
        EXPORTING
          value = wa_cliente-pstlz
          name  = `CODPOSTAL`.
      CALL METHOD lo_el_zwda_cliente->set_attribute
        EXPORTING
          value = wa_cliente-land1
          name  = `PAIS`.

* 2 -> obtener pedidos en curso para el cliente
      DATA: lit_ped_cliente TYPE STANDARD TABLE OF zwda_ped_cliente,
            wa_lit_ped_cliente TYPE zwda_ped_cliente.

      SELECT * FROM zwda_ped_cliente
      INTO CORRESPONDING FIELDS OF TABLE lit_ped_cliente
      WHERE cliente = wa_usuario-cliente
        AND funcion = 'RG'
        AND idioma = sy-langu.

      DATA lit_pedidos TYPE STANDARD TABLE OF zwda_pedido.
      DATA wa_pedidos  TYPE zwda_pedido.
```

```
      LOOP AT lit_ped_cliente INTO wa_lit_ped_cliente.
        MOVE-CORRESPONDING wa_lit_ped_cliente TO wa_pedidos.
        IF wa_pedidos-status = 'C'.
        ELSE.
        ENDIF.
        CALL FUNCTION 'CONVERSION_EXIT_ALPHA_OUTPUT'
          EXPORTING
            input  = wa_pedidos-fecha
          IMPORTING
            output = wa_pedidos-fecha.

        APPEND wa_pedidos TO lit_pedidos.
      ENDLOOP.

* 3 -> binding de la tabla lit_pedidos
      DATA lo_nd_zwda_pedido TYPE REF TO if_wd_context_node.
      DATA lo_el_zwda_pedido TYPE REF TO if_wd_context_element.
      DATA ls_zwda_pedido TYPE wd_this->element_zwda_pedido.
*   navigate from <CONTEXT> to <ZWDA_PEDIDO> via lead selection
      lo_nd_zwda_pedido = wd_context->get_child_node( name = wd_this-
>wdctx_zwda_pedido ).

* subtotales
      DATA: lt_field TYPE salv_wd_t_field_ref.
      DATA: ls_field LIKE LINE OF lt_field.
      DATA : lr_sort_rule TYPE REF TO cl_salv_wd_sort_rule.
      DATA : lr_aggr_rule TYPE REF TO cl_salv_wd_aggr_rule.
      DATA: l_alv_pedidos TYPE REF TO cl_salv_wd_config_table.
      DATA: l_ref_interfacecontroller TYPE REF TO iwci_salv_wd_table .

      l_ref_interfacecontroller = wd_this->wd_cpifc_alv_pedidos( ).
      l_alv_pedidos = l_ref_interfacecontroller->get_model( ).

      CALL METHOD l_alv_pedidos->if_salv_wd_field_settings~get_fields
        RECEIVING
          value = lt_field.

      READ TABLE lt_field INTO ls_field WITH KEY fieldname = 'PEDIDO'.

      CALL METHOD ls_field-r_field->if_salv_wd_sort~create_sort_rule
        EXPORTING
          group_aggregation = abap_true
        RECEIVING
          value             = lr_sort_rule.

      CLEAR ls_field.
      LOOP AT lt_field INTO ls_field.
        CASE ls_field-fieldname.
          WHEN 'PRECIO_TOT' OR 'PEDIDO'.
            CALL METHOD ls_field-r_field->if_salv_wd_aggr~create_aggr_rule
              RECEIVING
                value = lr_aggr_rule.
          WHEN OTHERS.
        ENDCASE.
      ENDLOOP.

*   @TODO handle not set lead selection
      IF NOT lo_nd_zwda_pedido IS INITIAL.
        CALL METHOD lo_nd_zwda_pedido->bind_table
          EXPORTING
            new_items = lit_pedidos.
      ENDIF.
ENDMETHOD.
```

ONACTIONCOMPRAR

Este método añade el artículo seleccionado por el usuario al pedido en curso.

```
METHOD onactioncomprar.

  DATA lo_nd_zwda_comprar TYPE REF TO if_wd_context_node.
  DATA lo_el_zwda_comprar TYPE REF TO if_wd_context_element.
  DATA ls_zwda_comprar TYPE wd_this->element_zwda_comprar.
  DATA lit_zwda_comprar TYPE STANDARD TABLE OF wd_this->element_zwda_comprar.
  DATA lo_nd_zwda_comprar_aux TYPE REF TO if_wd_context_node.
  DATA lo_el_zwda_comprar_aux TYPE REF TO if_wd_context_element.
  DATA ls_zwda_comprar_aux TYPE wd_this->element_zwda_comprar_aux.
* navigate from <CONTEXT> to <ZWDA_COMPRAR_AUX> via lead selection
  lo_nd_zwda_comprar_aux = wd_context->get_child_node( name = wd_this-
>wdctx_zwda_comprar_aux ).
  lo_el_zwda_comprar_aux = lo_nd_zwda_comprar_aux->get_element(  ).
  lo_el_zwda_comprar_aux->get_static_attributes(
    IMPORTING
      static_attributes = ls_zwda_comprar_aux ).
* añadir elemento al contexto
  lo_nd_zwda_comprar = wd_context->get_child_node( name = wd_this-
>wdctx_zwda_comprar ).
* @TODO handle not set lead selection
  IF NOT lo_nd_zwda_comprar IS INITIAL.

    CALL METHOD lo_nd_zwda_comprar->get_static_attributes_table
      IMPORTING
        table = lit_zwda_comprar.
* buscar precio
    wd_this->buscar_precio(
      EXPORTING
        material =  ls_zwda_comprar_aux-material
      IMPORTING
        precio =    ls_zwda_comprar_aux-precio_uni
    ).
* multiplicar precio unitario por cantidad
    ls_zwda_comprar = ls_zwda_comprar_aux.
    ls_zwda_comprar-precio_tot = ls_zwda_comprar_aux-
cantidad * ls_zwda_comprar_aux-precio_uni.
    APPEND ls_zwda_comprar TO lit_zwda_comprar.

ls_zwda_comprar-precio_tot = ls_zwda_comprar_aux-
precio_uni * ls_zwda_comprar_aux-cantidad.
    CALL METHOD lo_nd_zwda_comprar->bind_table
      EXPORTING
        new_items = lit_zwda_comprar.

    CALL METHOD lo_el_zwda_comprar_aux->set_static_attributes_null.
  ENDIF.
ENDMETHOD.
```

ONACTIONTORNAR

Borramos los datos y regresamos a la vista principal disparando el conector de salida.

```
METHOD onactiontornar.

* limpiar estructuras
  DATA lo_nd_zwda_pedido TYPE REF TO if_wd_context_node.
  DATA lo_el_zwda_pedido TYPE REF TO if_wd_context_element.
  DATA ls_zwda_pedido TYPE wd_this->element_zwda_pedido.
*   navigate from <CONTEXT> to <ZWDA_PEDIDO> via lead selection
  lo_nd_zwda_pedido = wd_context->get_child_node( name = wd_this-
>wdctx_zwda_pedido ).
  IF NOT lo_nd_zwda_pedido IS INITIAL.
```

```
* get element via lead selection
    lo_el_zwda_pedido = lo_nd_zwda_pedido->get_element(  ).
    IF NOT  lo_el_zwda_pedido IS INITIAL.
       CALL METHOD lo_nd_zwda_pedido->invalidate.
    ENDIF.
  ENDIF.

  DATA lo_nd_zwda_comprar TYPE REF TO if_wd_context_node.
  DATA lo_el_zwda_comprar TYPE REF TO if_wd_context_element.
  DATA ls_zwda_comprar TYPE wd_this->element_zwda_comprar.
*   navigate from <CONTEXT> to <ZWDA_COMPRAR> via lead selection
  lo_nd_zwda_comprar = wd_context->get_child_node( name = wd_this-
>wdctx_zwda_comprar ).
  IF NOT lo_nd_zwda_comprar IS INITIAL.
    lo_el_zwda_comprar = lo_nd_zwda_comprar->get_element(  ).
    IF NOT  lo_el_zwda_comprar IS INITIAL.

*      CALL METHOD lo_nd_zwda_comprar->set_static_attributes_null.
      CALL METHOD lo_nd_zwda_comprar->invalidate.

    ENDIF.
  ENDIF.
  wd_this->fire_back_main_plg( ).

ENDMETHOD.
```

ON_FUNCTION_ALV_COMPRAR

Ejecutamos la operación que el usuario ha seleccionado.

```
METHOD on_function_alv_comprar.

  CASE  r_param->id.
* HACER EL PEDIDO EN SAP
    WHEN 'TRANSFERIR'.
        wd_this->realizar_pedido(
        ).

* BORRAR UN MATERIAL del pedido
    WHEN 'BORRAR'.
      wd_this->borrar_item(
      ).
  ENDCASE.

ENDMETHOD.
```

BORRAR_ITEM

Eliminamos el producto seleccionado del pedido en curso.

```
METHOD borrar_item.

  DATA lo_nd_zwda_comprar TYPE REF TO if_wd_context_node.
  DATA lo_el_zwda_comprar TYPE REF TO if_wd_context_element.
  DATA ls_zwda_comprar TYPE wd_this->element_zwda_comprar.
  DATA lit_comprar  TYPE STANDARD TABLE OF wd_this->element_zwda_comprar.

*     navigate from <CONTEXT> to <ZWDA_PEDIDO> via lead selection
  lo_nd_zwda_comprar = wd_context->get_child_node( name = wd_this-
>wdctx_zwda_comprar ).

*     @TODO handle not set lead selection
  IF NOT lo_nd_zwda_comprar IS INITIAL.
*     get element via lead selection
```

```
    lo_el_zwda_comprar = lo_nd_zwda_comprar->get_element(  ).

*     @TODO handle not set lead selection
    IF NOT lo_el_zwda_comprar IS INITIAL.

*     get all declared attributes
      lo_el_zwda_comprar->get_static_attributes(
        IMPORTING
          static_attributes = ls_zwda_comprar ).


      CALL METHOD lo_nd_zwda_comprar->get_static_attributes_table
        IMPORTING
          table = lit_comprar.

      READ TABLE lit_comprar WITH KEY material = ls_zwda_comprar-material
      TRANSPORTING NO FIELDS.

      IF sy-subrc = 0.
        DELETE  lit_comprar INDEX sy-tabix.
        CALL METHOD lo_nd_zwda_comprar->bind_table
          EXPORTING
            new_items            = lit_comprar
      ENDIF.
    ENDIF.
  ENDIF.
ENDMETHOD.
```

BUSCAR_PRECIO

Recuperamos el precio de SAP del producto seleccionado.

```
METHOD buscar_precio .
  SELECT SINGLE stprs  FROM mbew
  INTO (precio)
  WHERE matnr = material.
ENDMETHOD.
```

REALIZAR_PEDIDO

Realizamos el pedido en SAP bajo la organización, canal de distribución y sector determinados.

```
METHOD realizar_pedido.

  DATA lo_api_controller     TYPE REF TO if_wd_controller.
  DATA lo_message_manager    TYPE REF TO if_wd_message_manager.
  DATA lv_cliente TYPE bdc_fval, lv_fecha TYPE bdc_fval.
  DATA lit_messtab TYPE STANDARD TABLE OF  bdcmsgcoll.
  DATA wa_messtab  TYPE bdcmsgcoll, l_error     TYPE sy-subrc.
  DATA lo_nd_zwda_comprar TYPE REF TO if_wd_context_node.
  DATA lit_comprar TYPE STANDARD TABLE OF zwda_pedido.
  DATA lo_nd_zusuario TYPE REF TO if_wd_context_node.
  DATA lo_el_zusuario TYPE REF TO if_wd_context_element.
  DATA ls_zusuario TYPE wd_this->element_zusuario.
  DATA lv_cliente_aux LIKE ls_zusuario-cliente.
  lv_cliente = lv_cliente_aux.
  lo_api_controller ?= wd_this->wd_get_api( ).

  CALL METHOD lo_api_controller->get_message_manager
    RECEIVING
      message_manager = lo_message_manager.
```

```
  lo_nd_zusuario = wd_context->get_child_node( name = wd_this-
>wdctx_zusuario ).

  lo_el_zusuario = lo_nd_zusuario->get_element(  ).
  lo_el_zusuario->get_attribute(
    EXPORTING
      name =  `CLIENTE`
    IMPORTING
      value = lv_cliente_aux ).

  lo_nd_zwda_comprar = wd_context->get_child_node( name = wd_this-
>wdctx_zwda_comprar ).

  IF NOT lo_nd_zwda_comprar  IS INITIAL.

    CALL METHOD lo_nd_zwda_comprar->get_static_attributes_table
      IMPORTING
        table = lit_comprar.

    IF NOT lit_comprar IS INITIAL.

      DATA lfecha_aux TYPE bdc_fval.
      lv_fecha = sy-datum.
      CONCATENATE lv_fecha+6(2) '.'              INTO lfecha_aux.
      CONCATENATE lfecha_aux   lv_fecha+4(2) '.' INTO lfecha_aux.
      CONCATENATE lfecha_aux   lv_fecha(4)       INTO lfecha_aux.
* centro WERKS  -> 1000 Berlin
* orgenizacion de ventas VKORG -> 1000
* canal de distribucuión VTWEG -> 10
      CALL FUNCTION 'ZWDA_PEDIDO'
        EXPORTING
*   AUART_001         = 'ZDAN'
*   VKORG_002         = '1000'
*   VTWEG_003         = '10'
*   SPART_004         = '00'
          kunnr_005        = lv_cliente
          kunnr_006        = lv_cliente
          ketdat_007       = lfecha_aux
          prsdt_009        = lfecha_aux
          kunnr_014        = lv_cliente
          kunnr_015        = lv_cliente
          ketdat_016       = lfecha_aux
          prsdt_018        = lfecha_aux
        IMPORTING
          subrc            = l_error
        TABLES
         messtab         = lit_messtab
         it_pedido       = lit_comprar.
      CALL METHOD wd_this->reportar_errores().
      CALL METHOD wd_this->linkar_tabla().
  ENDIF.
  ENDIF.
ENDMETHOD.
```

REPORTAR_ERRORES

Reportamos los mensajes que ha generado el proceso de creación de pedido de SAP.

```
METHOD reportar_errores.

 READ TABLE lit_messtab WITH KEY msgid = 'V1 msgnr = '311
 msgtyp = 'S' msgspra = sy-langu INTO wa_messtab.
 IF sy-subrc = 0.
 DATA lsyst TYPE syst.
*       report message
 lsyst-msgv1 = wa_messtab-msgv1.
```

```
 lsyst-msgv2 = wa_messtab-msgv2.
 lsyst-msgv3 = wa_messtab-msgv3.

 CALL METHOD lo_message_manager->report_t100_message
   EXPORTING
     msgid = wa_messtab-msgid
     msgno = '311'
     msgty = 'S'
     p1    = lsyst-msgv1
     p2    = lsyst-msgv2
     p3    = lsyst-msgv3.
ENDMETHOD.
```

LINKAR_PEDIDOS

Actualizamos las tablas para que muestren el nuevo pedido que se acaba de generar

```
METHOD linkar_pedidos.

* binding del nuevo pedido
   DATA wa_pedidos TYPE zwda_pedido.
  DATA lo_nd_zwda_pedido TYPE REF TO if_wd_context_node.
  DATA lo_el_zwda_pedido TYPE REF TO if_wd_context_element.
  DATA ls_zwda_pedido TYPE wd_this->element_zwda_pedido.
  DATA lit_pedidos TYPE STANDARD TABLE OF zwda_pedido.
lo_nd_zwda_pedido = wd_context->get_child_node( name = wd_this-
>wdctx_zwda_pedido ).
* @TODO handle not set lead selection
  IF NOT lo_nd_zwda_pedido IS INITIAL.
    CALL METHOD lo_nd_zwda_pedido->get_static_attributes_table
      IMPORTING
        table = lit_pedidos.

    LOOP AT lit_comprar INTO wa_pedidos.
      wa_pedidos-pedido = lsyst-msgv2.
      APPEND wa_pedidos TO lit_pedidos.
    ENDLOOP.
    CALL METHOD lo_nd_zwda_pedido->bind_table
      EXPORTING
       new_items = lit_pedidos.
    REFRESH lit_comprar.
    CALL METHOD lo_nd_zwda_comprar->bind_table
      EXPORTING
       new_items = lit_comprar.
  ENDIF.
ENDMETHOD.
```

ZWDA_PEDIDO

Ejecutamos el proceso de creación del pedido.

```
FUNCTION zwda_pedido.

  DATA pos TYPE string.
* items del pedido
  DATA wa_pedido TYPE zwda_pedido.
  DATA num TYPE i.
  subrc = 0.

  PERFORM bdc_nodata      USING nodata.

  PERFORM open_group      USING group user keep holddate ctu.
```

```
  PERFORM bdc_dynpro      USING 'SAPMV45A' '0101'.
  PERFORM bdc_field       USING 'BDC_CURSOR'
                                'VBAK-AUART'.
  PERFORM bdc_field       USING 'BDC_OKCODE'
                                '/00'.
* tipo de pedido
  PERFORM bdc_field       USING 'VBAK-AUART'
                                auart_001.
* organizacion de ventas
  PERFORM bdc_field       USING 'VBAK-VKORG'
                                vkorg_002.
* canal
  PERFORM bdc_field       USING 'VBAK-VTWEG'
                                vtweg_003.
* sector
  PERFORM bdc_field       USING 'VBAK-SPART'
                                spart_004.

  LOOP AT it_pedido INTO wa_pedido.
    pos = sy-tabix.
    num = STRLEN( pos ).
    num = num - 1.
    pos = pos(num).
* material
    PERFORM fill_dynpro_4001 USING wa_pedido
                                   pos
                                   kunnr_005
                                   ketdat_007.
    CLEAR pos.
  ENDLOOP.
  PERFORM fill_dynpro_4001_b USING kunnr_014
                                   ketdat_016 .
  PERFORM bdc_dynpro      USING 'SAPLSPO2' '0101'.
  PERFORM bdc_field       USING 'BDC_OKCODE'
                                '=OPT1'.
  PERFORM bdc_transaction TABLES messtab
  USING                         'VA01' ctu
                                mode
                                update.
  IF sy-subrc <> 0.
    subrc = sy-subrc.
    EXIT.
  ENDIF.

  PERFORM close_group USING     ctu.

ENDFUNCTION.
```

WDOMODIFYVIEW

Este evento se ejecuta cada vez que el usuario interactúa con la vista. Recupera de SAP datos del cliente para poder realizar el pedido y los pedidos pendientes de entrega.

```
METHOD wddomodifyview.

  DATA lo_nd_zusuario TYPE REF TO if_wd_context_node.
  DATA lo_el_zusuario TYPE REF TO if_wd_context_element.
  DATA ls_zusuario TYPE wd_this->element_zusuario.
  DATA lv_cliente LIKE ls_zusuario-cliente.
  DATA lo_nd_zwda_cliente TYPE REF TO if_wd_context_node.
  DATA lo_el_zwda_cliente TYPE REF TO if_wd_context_element.
  DATA ls_zwda_cliente TYPE wd_this->element_zwda_cliente.
```

```
* navigate from <CONTEXT> to <ZUSUARIO> via lead selection
  lo_nd_zusuario = wd_context->get_child_node( name = wd_this-
>wdctx_zusuario ).

* get element via lead selection
  lo_el_zusuario = lo_nd_zusuario->get_element(  ).

* get single attribute
  lo_el_zusuario->get_attribute(
    EXPORTING
      name =  `CLIENTE`
    IMPORTING
      value = lv_cliente ).
* binding cliente
  DATA wa_cliente TYPE kna1.
  SELECT SINGLE * FROM kna1 INTO wa_cliente WHERE kunnr = lv_cliente.
* navigate from <CONTEXT> to <ZWDA_CLIENTE> via lead selection
  lo_nd_zwda_cliente = wd_context->get_child_node( name = wd_this-
>wdctx_zwda_cliente ).
* get element via lead selection
  lo_el_zwda_cliente = lo_nd_zwda_cliente->get_element(  ).

  CALL METHOD lo_el_zwda_cliente->set_attribute
    EXPORTING
      value = wa_cliente-name1
      name  = `NOMBRE`.
  CALL METHOD lo_el_zwda_cliente->set_attribute
    EXPORTING
      value = wa_cliente-stras
      name  = `CALLE`.
  CALL METHOD lo_el_zwda_cliente->set_attribute
    EXPORTING
      value = wa_cliente-ort01
      name  = `POBLACION`.
  CALL METHOD lo_el_zwda_cliente->set_attribute
    EXPORTING
      value = wa_cliente-telf1
      name  = `TELEFONO`.
  CALL METHOD lo_el_zwda_cliente->set_attribute
    EXPORTING
      value = wa_cliente-ort01
      name  = `CIUDAD`.
  CALL METHOD lo_el_zwda_cliente->set_attribute
    EXPORTING
      value = wa_cliente-pstlz
      name  = `CODPOSTAL`.
  CALL METHOD lo_el_zwda_cliente->set_attribute
    EXPORTING
      value = wa_cliente-land1
      name  = `PAIS`.

* 2 -> obtener pedidos en curso para el cliente
  DATA: lit_ped_cliente TYPE STANDARD TABLE OF zwda_ped_cliente,
        wa_lit_ped_cliente TYPE zwda_ped_cliente.

  SELECT * FROM zwda_ped_cliente
  INTO CORRESPONDING FIELDS OF TABLE lit_ped_cliente
  WHERE cliente = lv_cliente AND funcion = 'RG' AND idioma = sy-langu.

  DATA lit_pedidos TYPE STANDARD TABLE OF zwda_pedido.
  DATA wa_pedidos  TYPE zwda_pedido.

  LOOP AT lit_ped_cliente INTO wa_lit_ped_cliente.
    MOVE-CORRESPONDING wa_lit_ped_cliente TO wa_pedidos.
    clear wa_pedidos-text.
    if wa_pedidos-status = 'C'.
      wa_pedidos-text = 'Completely processed'.
```

```
    elseif wa_pedidos-status = 'B'.
      wa_pedidos-text = 'Partially processed'.
    elseif wa_pedidos-status = 'A'.
      wa_pedidos-text = 'Not yet processed'.
    else.
       wa_pedidos-text = 'Not Relevant'.
    endif.
    CALL FUNCTION 'CONVERSION_EXIT_PDATE_OUTPUT'
      EXPORTING
        input  = wa_pedidos-fecha
      IMPORTING
        output = wa_pedidos-fecha.
    APPEND wa_pedidos TO lit_pedidos.
  ENDLOOP.

* 3 -> binding de la tabla lit_pedidos
  DATA lo_nd_zwda_pedido TYPE REF TO if_wd_context_node.
  DATA lo_el_zwda_pedido TYPE REF TO if_wd_context_element.
  DATA ls_zwda_pedido TYPE wd_this->element_zwda_pedido.
*   navigate from <CONTEXT> to <ZWDA_PEDIDO> via lead selection
  lo_nd_zwda_pedido = wd_context->get_child_node( name = wd_this-
>wdctx_zwda_pedido ).

* subtotales
  DATA: lt_field TYPE salv_wd_t_field_ref.
  DATA: ls_field LIKE LINE OF lt_field.
  DATA : lr_sort_rule TYPE REF TO cl_salv_wd_sort_rule.
  DATA : lr_aggr_rule TYPE REF TO cl_salv_wd_aggr_rule.
  DATA: l_alv_pedidos TYPE REF TO cl_salv_wd_config_table.
  DATA: l_ref_interfacecontroller TYPE REF TO iwci_salv_wd_table .
  l_ref_interfacecontroller = wd_this->wd_cpifc_alv_pedidos( ).

  l_alv_pedidos = l_ref_interfacecontroller->get_model( ).
  CALL METHOD l_alv_pedidos->if_salv_wd_field_settings~get_fields
    RECEIVING
      value = lt_field.

  READ TABLE lt_field INTO ls_field WITH KEY fieldname = 'PEDIDO'.
  CALL METHOD ls_field-r_field->if_salv_wd_sort~create_sort_rule
    EXPORTING
      group_aggregation = abap_true
    RECEIVING
      value             = lr_sort_rule.
  CLEAR ls_field.
  LOOP AT lt_field INTO ls_field.
    CASE ls_field-fieldname.
      WHEN 'PRECIO_TOT' OR 'PEDIDO'.
        CALL METHOD ls_field-r_field->if_salv_wd_aggr~create_aggr_rule
          RECEIVING
            value = lr_aggr_rule.
      WHEN OTHERS.
    ENDCASE.
  ENDLOOP.
*   @TODO handle not set lead selection
  IF NOT lo_nd_zwda_pedido IS INITIAL.
    CALL METHOD lo_nd_zwda_pedido->bind_table
      EXPORTING
        new_items = lit_pedidos.
  ENDIF.

ENDMETHOD.
```

7.7 El controlador vista principal

7.7.1 Contexto

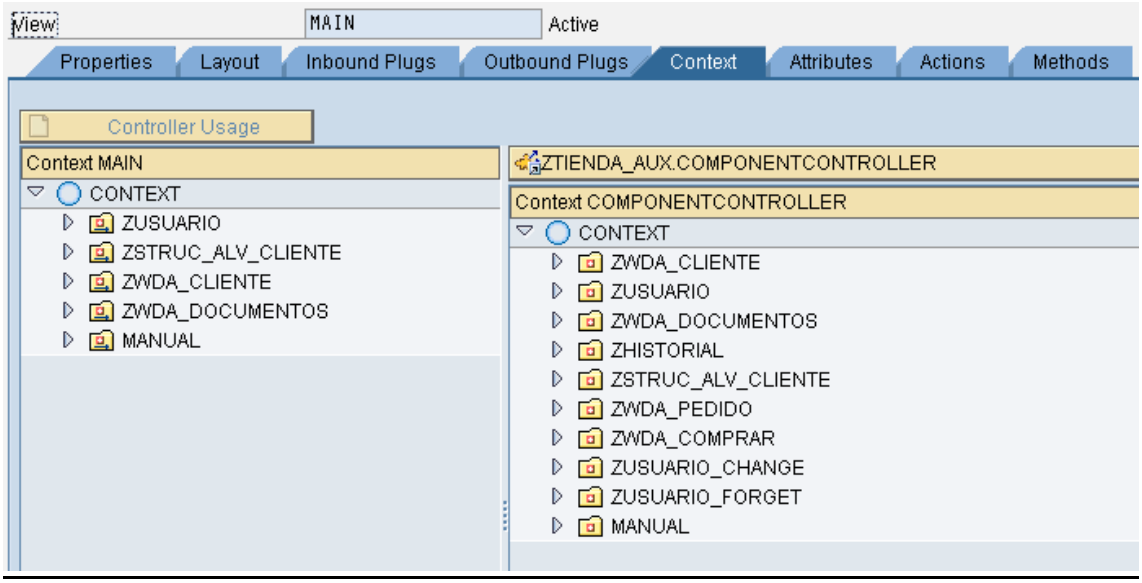


Figura 27: contexto controlador principal

7.7.2 Vista

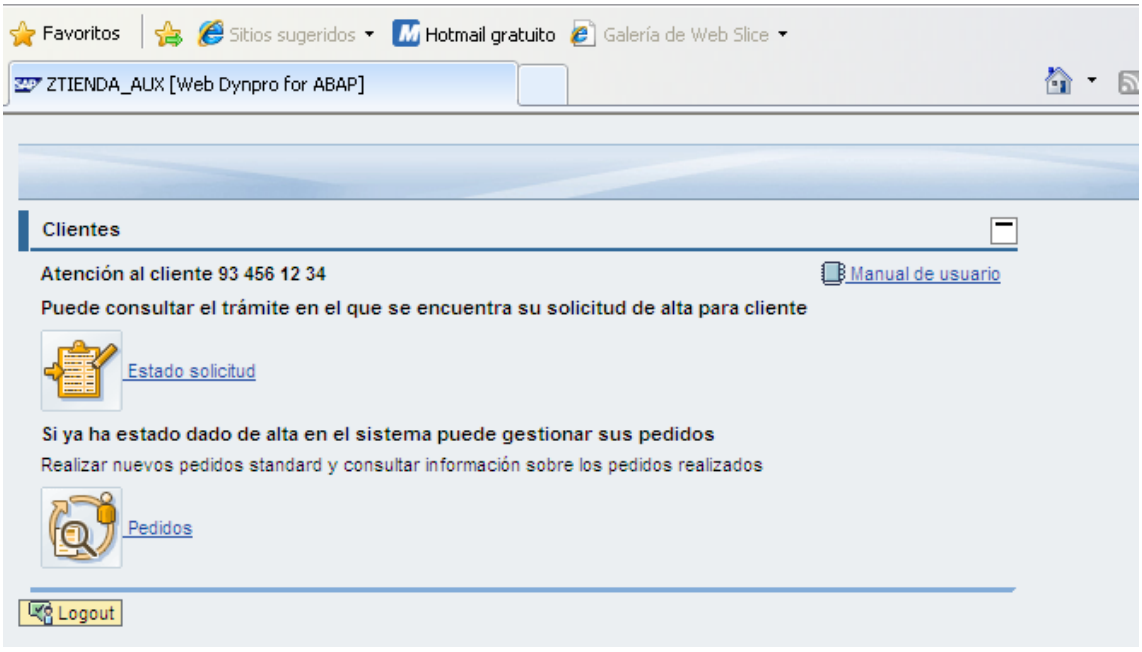


Figura 28: interfaz de usuario

7.7.2.1 Conectores entrada

IN_MAIN→entrada a la vista gestión de pedidos.

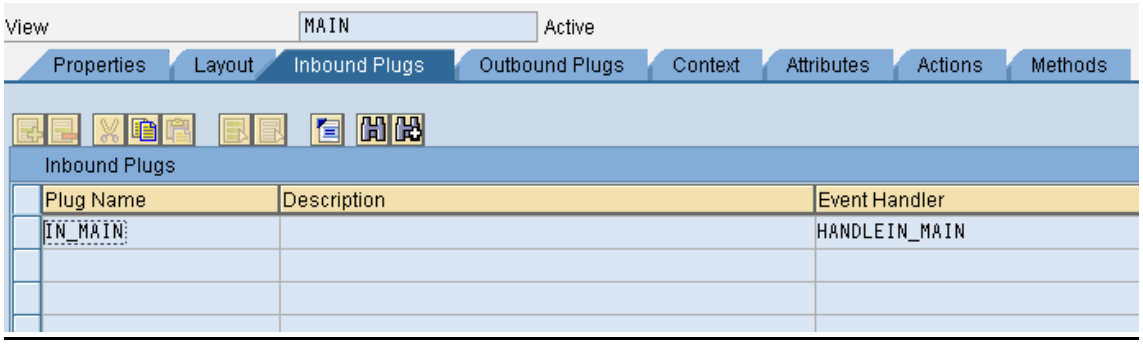


Figura 29: conector entrada

7.5.2.1 Conectores salida

OUT_AUTENTIFICACIO→regreso a la vista autentificación.

OUT_CON_CLIENTES →navegar hacia gestión de clientes.

OUT_CON_PEDIDOS →navegar hacia gestión de pedidos.

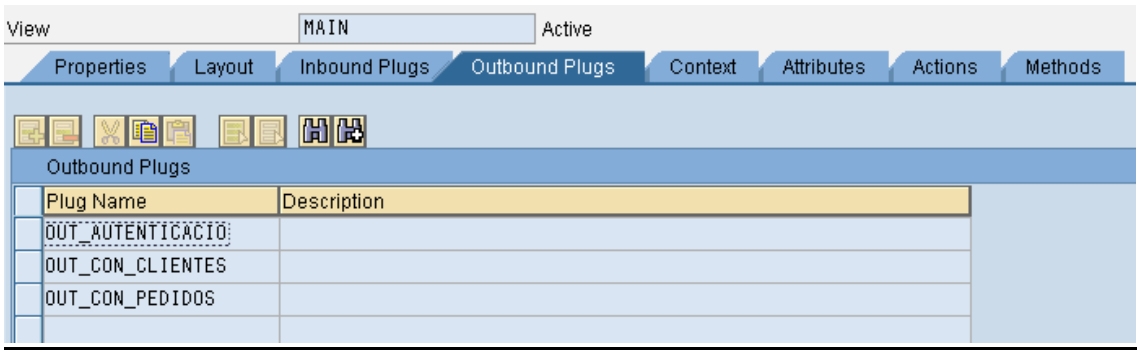


Figura 30: conectores salida

7.7.3 Métodos

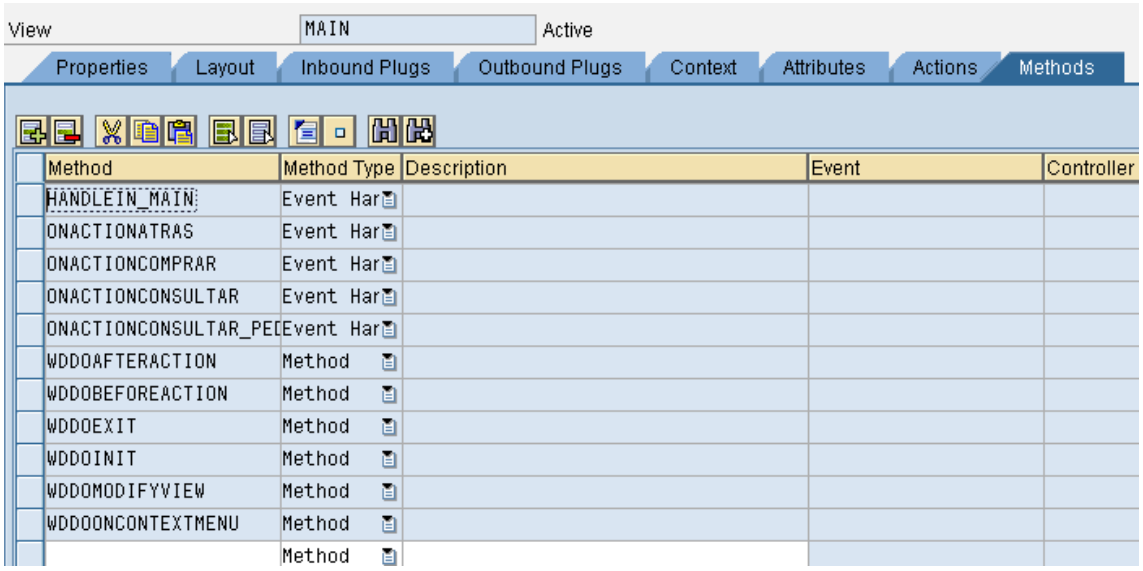


Figura 31: métodos

HANDLE_MAIN

Este evento se ejecuta cuando navegamos a la vista principal, se limpian las estructuras y se carga el manual. Es el evento asociado al conector de entrada IN_MAIN

```
METHOD handlein_main.

  DATA lo_nd_zwda_cliente TYPE REF TO if_wd_context_node.
  DATA lo_el_zwda_cliente TYPE REF TO if_wd_context_element.
  DATA ls_zwda_cliente TYPE wd_this->element_zwda_cliente.
* navigate from <CONTEXT> to <ZWDA_CLIENTE> via lead selection
  lo_nd_zwda_cliente = wd_context->get_child_node( name = wd_this-
>wdctx_zwda_cliente ).
  IF NOT lo_nd_zwda_cliente IS INITIAL.

* get element via lead selection
    lo_el_zwda_cliente = lo_nd_zwda_cliente->get_element(  ).
*    IF NOT lo_el_zwda_cliente IS INITIAL.
    CALL METHOD lo_nd_zwda_cliente->set_static_attributes_null.
    CALL METHOD lo_nd_zwda_cliente->invalidate.
*    ENDIF.
  ENDIF.
*leer manual
  DATA wa_manual TYPE zwda_documentos.
  SELECT SINGLE * FROM zwda_documentos
    INTO wa_manual
    WHERE identificador = '*'.

  DATA lo_nd_manual TYPE REF TO if_wd_context_node.
  DATA lo_el_manual TYPE REF TO if_wd_context_element.
  DATA ls_manual TYPE wd_this->element_manual.
* navigate from <CONTEXT> to <MANUAL> via lead selection
  lo_nd_manual = wd_context->get_child_node( name = wd_this-
>wdctx_manual ).
  MOVE-CORRESPONDING wa_manual TO ls_manual.
  CALL METHOD lo_nd_manual->set_static_attributes
    EXPORTING
      index             = 1
      static_attributes = ls_manual.

ENDMETHOD.
```

ONACTIONATRAS

Esta acción dispara el conector de salida OUT_AUTENTIFICACIO.

```
method ONACTIONATRAS .
* invalidar cliente
* usario
  DATA lo_nd_zusuario TYPE REF TO if_wd_context_node.
  DATA lo_el_zusuario TYPE REF TO if_wd_context_element.
  DATA ls_zusuario TYPE wd_this->element_zusuario.
* navigate from <CONTEXT> to <ZUSUARIO> via lead selection
  lo_nd_zusuario = wd_context->get_child_node( name = wd_this-
>wdctx_zusuario ).

* get element via lead selection
  lo_el_zusuario = lo_nd_zusuario->get_element(  ).
```

```
  CALL METHOD lo_nd_zusuario->set_static_attributes_null.

  wd_this->fire_out_autenticacio_plg(
    ).

endmethod.
```

ONACTIONCONSULTAR

Esta acción dispara el conector de salida OUT_CON_CLIENTES →navegar hacia gestión de clientes.

```
method ONACTIONCONSULTAR .
  wd_this->fire_out_con_clientes_plg(
  ).
endmethod.
```

ONACTIONCONSULTAR_PEDIDO

Esta acción se ejecuta cuando el usuario selecciona acceder a la gestión de pedidos. Si el usuario es empleado navegamos directamente a gestión de pedidos. En el caso de que el usuario no sea empleado debe darse la condición de que existe como cliente en SAP para poder navegar mediante el conector de salida OUT_CON_PEDIDOS →navegar hacia gestión de pedidos.

```
METHOD onactionconsultar_pedido.

  DATA l_messag TYPE string.
  DATA lo_nd_zusuario TYPE REF TO if_wd_context_node.
  DATA lo_el_zusuario TYPE REF TO if_wd_context_element.
  DATA ls_zusuario TYPE wd_this->element_zusuario.
  DATA lv_usuario LIKE ls_zusuario-usuario.
*   navigate from <CONTEXT> to <ZUSUARIO> via lead selection
  lo_nd_zusuario = wd_context->get_child_node( name = wd_this-
>wdctx_zusuario ).
*   get element via lead selection
  lo_el_zusuario = lo_nd_zusuario->get_element(  ).
*   get single attribute
  lo_el_zusuario->get_attribute(
    EXPORTING
      name =  `USUARIO`
    IMPORTING
      value = lv_usuario ).
  CALL FUNCTION 'CONVERSION_EXIT_ALPHA_INPUT'
    EXPORTING
      input  = lv_usuario
    IMPORTING
      output = lv_usuario.

  DATA wa_usuario TYPE zusuarios.
  SELECT SINGLE * INTO wa_usuario
  FROM zusuarios WHERE usuario =  lv_usuario.
```

```
* comprobar si es cliente
  IF wa_usuario-es_empleado = 'X' OR wa_usuario-
cliente IS NOT INITIAL.
    wd_this->fire_out_con_pedidos_plg(
    ).
  ELSE.
    DATA lo_message_manager    TYPE REF TO if_wd_message_manager.
    DATA l_view_controller TYPE REF TO if_wd_view_controller.
    l_view_controller = wd_this->wd_get_api( ).
    CALL METHOD l_view_controller->get_message_manager
      RECEIVING
        message_manager = lo_message_manager.
    l_messag = 'No es cliente en el sistema ERP'.
    CALL METHOD lo_message_manager->raise_error_message
      EXPORTING
        message_text = l_messag.

  ENDIF.
ENDMETHOD.
```

8. Pruebas

Para garantizar que un software es correcto es necesario elaborar un plan de pruebas que garantice la calidad del producto desarrollado.

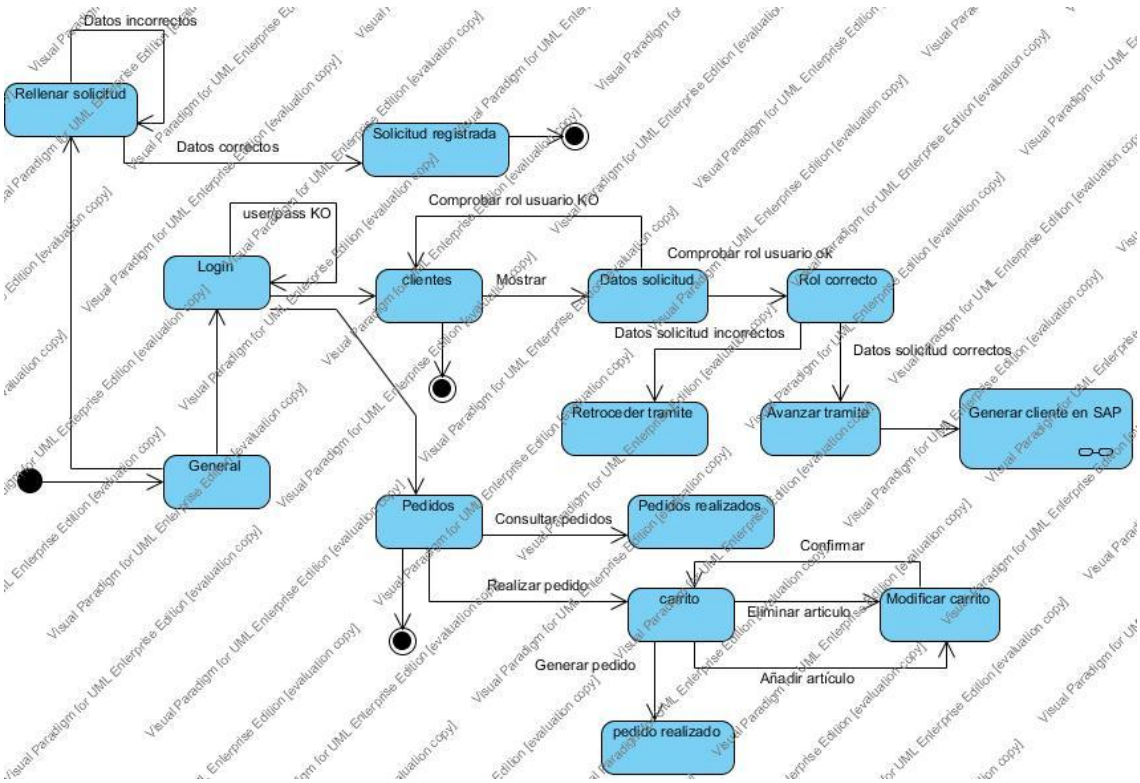
Las pruebas realizadas han sido unitarias durante todo el desarrollo del software. Este tipo de pruebas han garantizado que cada funcionalidad del sistema está libre de errores y ha permitido descartar errores frecuentes en la generación del código. De esta forma se obtiene una mínima calidad durante todo el desarrollo del software.

Sin embargo estas pruebas muy útiles durante la implementación no son suficientes porque no tienen en cuenta la integración con otros componentes. Por este motivo se ha realizado un juego de pruebas integrado. Este juego de pruebas garantizara que el sistema funciona bien desde un punto de vista global y no únicamente de funcionalidades individuales.

8.1 Plan de pruebas

El plan de pruebas realizado se ha basado en pruebas funcionales o de caja negra. En el contexto de las pruebas de sistemas orientados a objetos, (Kirani and Tsai 1994) consideran la *secuencia* como un concepto fundamental para las prueba de clases. Una secuencia representa el orden correcto en el que los métodos públicos de una clase pueden ser invocados. De acuerdo con estos autores, una de las aplicaciones de las secuencias es la prueba de clases a partir de su especificación como una máquina de estados. De esta forma puede utilizarse la especificación del comportamiento de una cierta clase en forma de máquina de estados para generar casos de prueba, ejecutarlos y medir nuevos criterios de cobertura.

El plan de pruebas se ha realizado basándose en el modelo de estados que se definió en la especificación:



La siguiente tabla muestra las distintas secuencias que se generan en el sistema y que se ha verificado con éxito:

1. Clase solicitud_cliente:

Secuencia	Métodos
Registrar solicitud en el sistema	
	Comprobar datos
	Registrar solicitud en el sistema
	Obtener claves
	Mostrar claves y mensaje confirmativo
Solicitud no registrada en el sistema	
	Comprobar datos incorrectos
	Lanzar mensaje datos incorrectos
	No registrar en el sistema
Validar solicitud	
	Autentificación
	Escoger solicitud

	Comprobar rol usuario
	Comprobar si es tramite final
	Comprobar si ya es cliente
	Avanzar trámite
Rechazar solicitud	
	Autentificación
	Escoger solicitud
	Obtener circuito activo
	Comprobar rol usuario
	Comprobar si es el primer trámite
	Comprobar si ya es cliente
	Retroceder trámite
Historial solicitud	
	Autentificación
	Escoger solicitud
	Comprobar rol usuario
	Visualizar solicitud
Generar cliente	
	Autentificación
	Obtener circuito activo
	Comprobar rol usuario
	Comprobar si es tramite final
	Comprobar si ya es cliente
	Generar cliente en SAP
Anexar documentación	
	Comprobar si el tipo mime lo soporta SAP
	Anexar documentación
Rechazo definitivo solicitud	
	Autentificación
	Comprobar rol usuario

	Rechazar definitivamente

2. Clase Pedido

Visualizar pedidos	
	Autentificación
	Comprobar rol usuario
	Visualizar pedidos
Añadir artículo pedido	
	Autentificación
	Comprobar rol usuario
	Escoger artículo
	Añadir artículo
Eliminar artículo pedido	
	Autentificación
	Comprobar rol usuario
	Escoger artículo
	Añadir artículo
Generar pedido	
	Autentificación
	Comprobar rol usuario
	Escoger artículo
	Añadir artículo

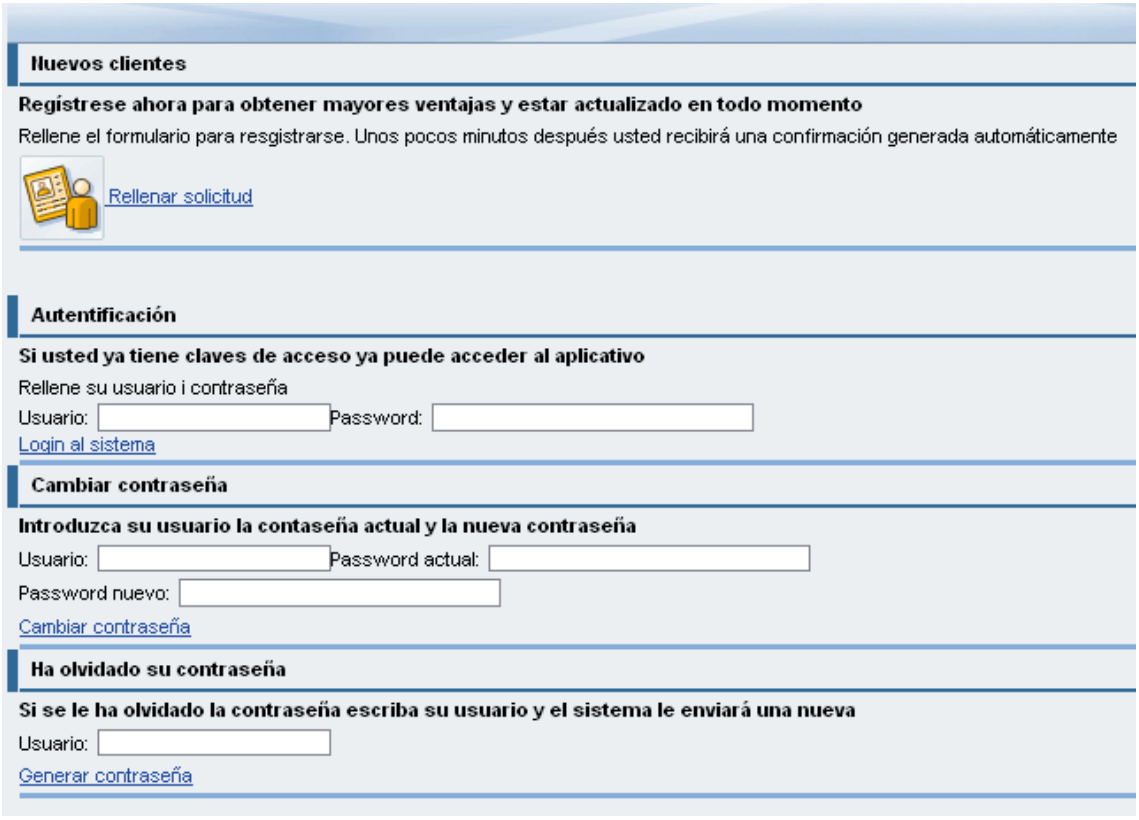
9. Manual de usuario

En este capítulo se explicara cómo debe usar el sistema el usuario cliente y el empleado.

9.1 Autentificación

Cuando el visitante accede por primera vez llegara a la página principal y tendrá dos opciones:

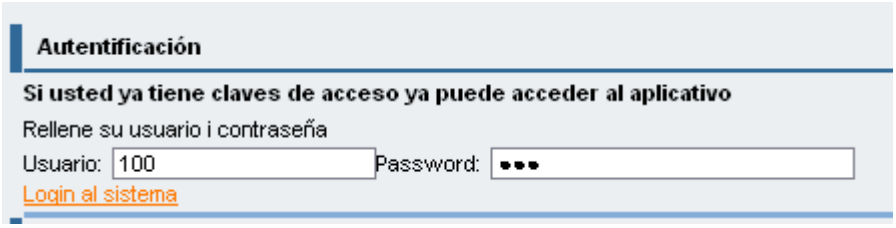
- autentificarse si previamente ya se ha registrado en el sistema
- registrarse en el sistema por primera vez:



Pantalla 1

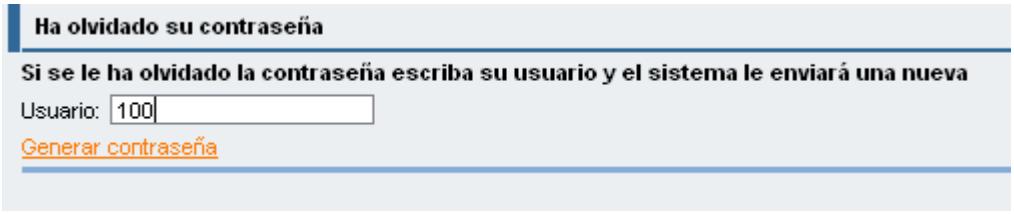
Si el visitante ya ha obtenido sus claves: usario y contraseña ya podrá navegar a la siguiente vista introduciendo dichas claves en los campos correspondientes y pulsando el link login al sistema.

La pantalla 2 muestra los campos que el usuario debe informar:



Pantalla 2

Si el usuario olvidó su contraseña este es el apartado correspondiente para pedirle al sistema que genere una nueva y esta se le enviará por e-mail:

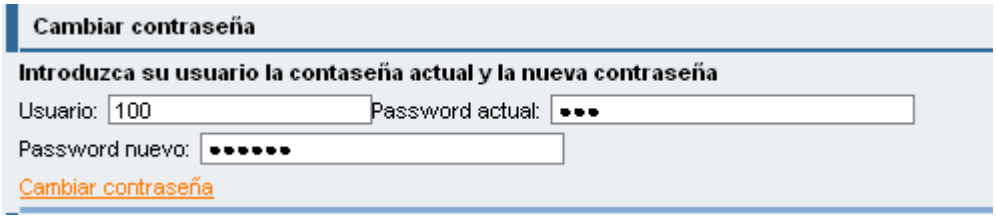


Pantalla 3

Si el usuario quiere cambiar la contraseña deberá introducir la siguiente información:

- usuario
- contraseña vieja
- contraseña nueva

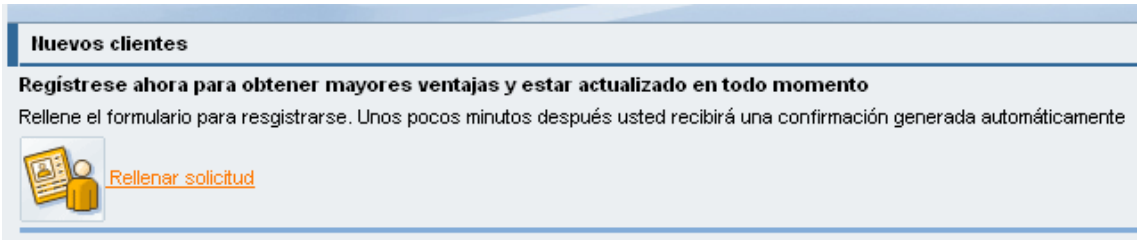
y pinchar el link "Cambiar contraseña".



Pantalla 4

Si por el contrario el visitante no tiene claves del sistema, es decir todavía no es usuario

Deberá rellenar el formulario con sus datos personales. A este formulario podrá acceder mediante el link “Rellenar formulario” en el apartado nuevos clientes:

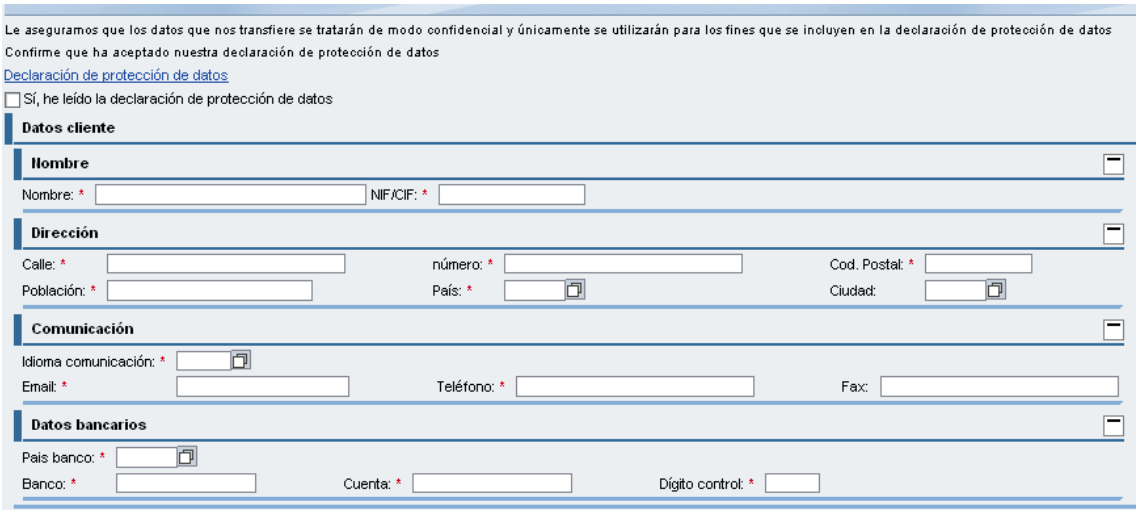


Pantalla 5

9.2 Formulario

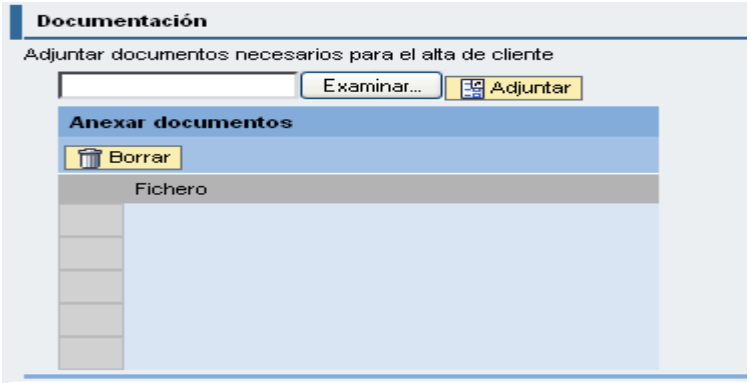
El formulario está formado por dos apartados los datos personales y la documentación asociada a los datos que son documentos que el visitante podrá anexar:

A continuación se muestra el apartado que hace referencia a los datos personales:



Pantalla 6

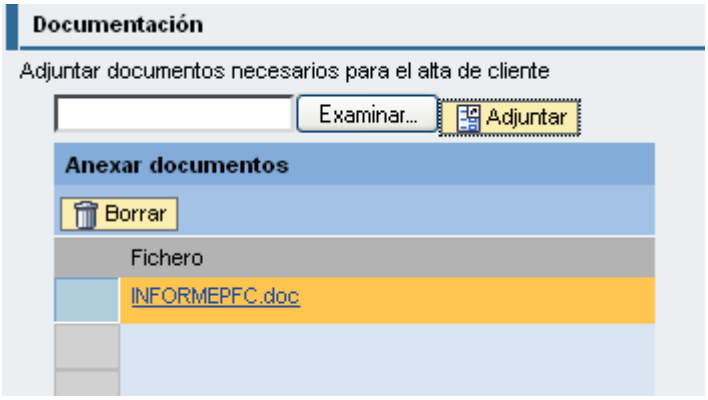
La documentación asociada al formulario se debe adjuntar en el siguiente apartado:



Pantalla 7

Para poder adjuntar documentación a la solicitud de cliente o formulario. El visitante deberá seleccionar un documento mediante el botón “examinar”. Después de

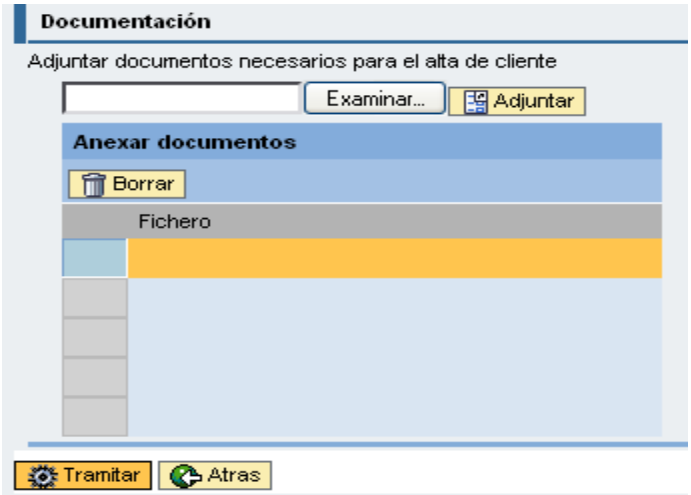
seleccionarlo deberá adjuntarlo mediante el botón “adjuntar”· Posteriormente el fichero se visualizara en la tabla:



Pantalla 8

Para poder ver el contenido de cualquier fichero anexado, únicamente se deberá pinchar sobre el nombre del fichero y el sistema operativo le pedirá confirmación para descargarlo.

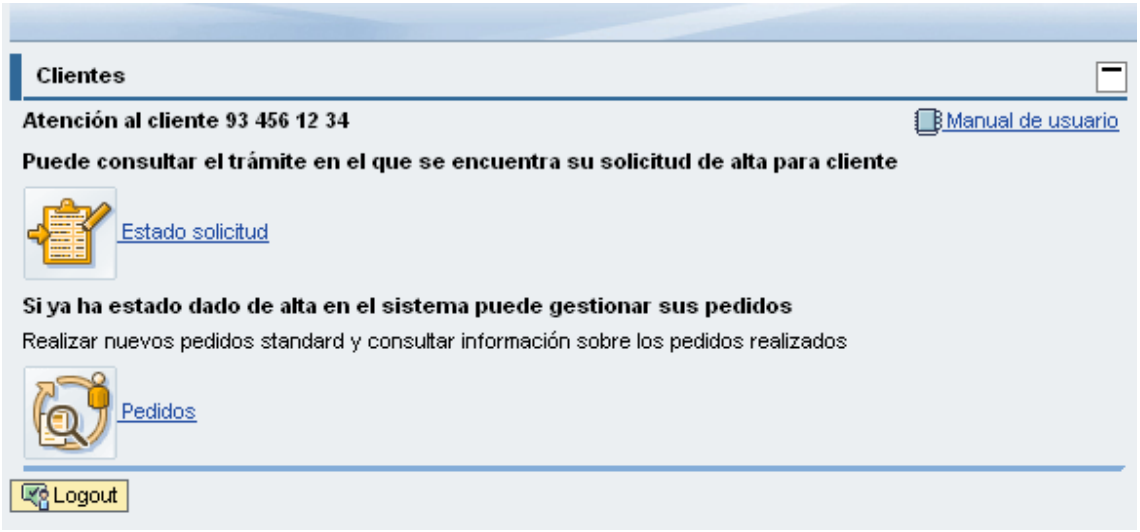
Para registrar el formulario en el sistema previamente deberá rellenar todos los datos ya mencionados y pulsar el botón “Tramitar”. El sistema le enviará las claves de acceso via e-mail pero también las mostrará por pantalla en el momento del registro. Mediante el botón atrás el visitante abandonará el formulario.



Pantalla 9

9.3 Vista principal

Si el usuario tiene claves de acceso y se ha autentificado con éxito navegará a la siguiente vista:

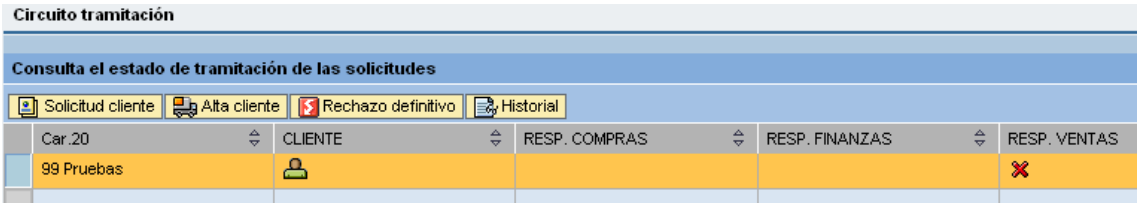


Pantalla 10

Desde esta vista podrá navegar a la gestión de clientes mediante el link "Estado de su solicitud" y a la gestión de pedidos mediante el otro link "Pedidos". Mediante el botón de logout el usuario abandona el sistema.

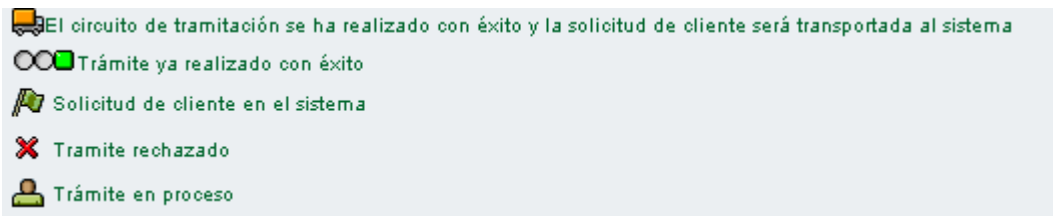
9.4 Gestión de clientes

En esta vista el usuario podrá consultar el estado de su solicitud de cliente:



Pantalla 11

La leyenda se muestra a continuación:



Pantalla 12

Cuando el usuario tenga bandera verde indicará que ya existe como cliente en SAP y podrá acceder a gestionar pedidos.

En caso de que el usuario sea un empleado deberá acceder a esta vista para validar los datos introducidos en los formularios. El usuario empleado visualizará todas las solicitudes de clientes que hay en el sistema:

Pantalla 13

Para poder validar una solicitud el usuario pulsará el botón “solicitud de cliente”, de esta forma navegará al formulario. En el formulario deberá pulsar el botón “tramitar” en el caso que la información introducida sea correcta y en caso contrario el botón “rechazo”.

Si el usuario no puede ejecutar la operación por falta de privilegios el sistema lanzará un mensaje de error.

Para ejecutar el alta de un cliente en SAP el usuario empleado deberá pulsar el botón “Alta cliente”

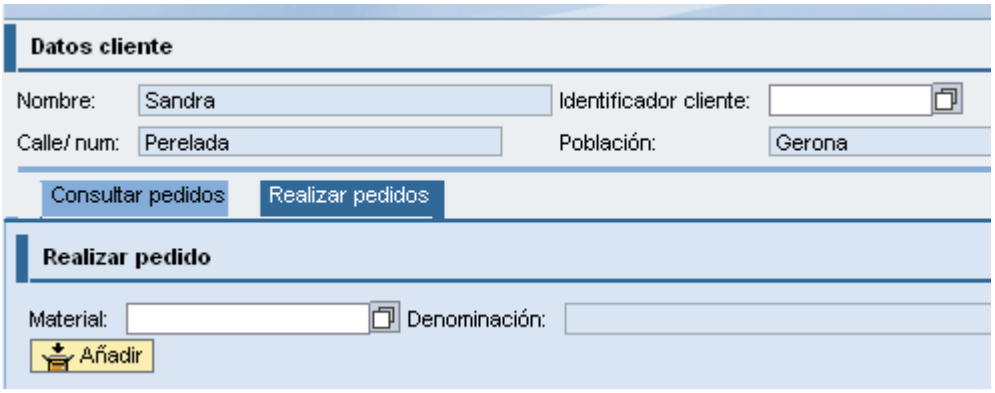
Para rechazar definitivamente una solicitud de cliente deberá pulsar el botón “Rechazo definitivo”.

El usuario podrá acceder al historial de la solicitud pulsando el botón “Historial”.

9.5 Gestión de pedidos

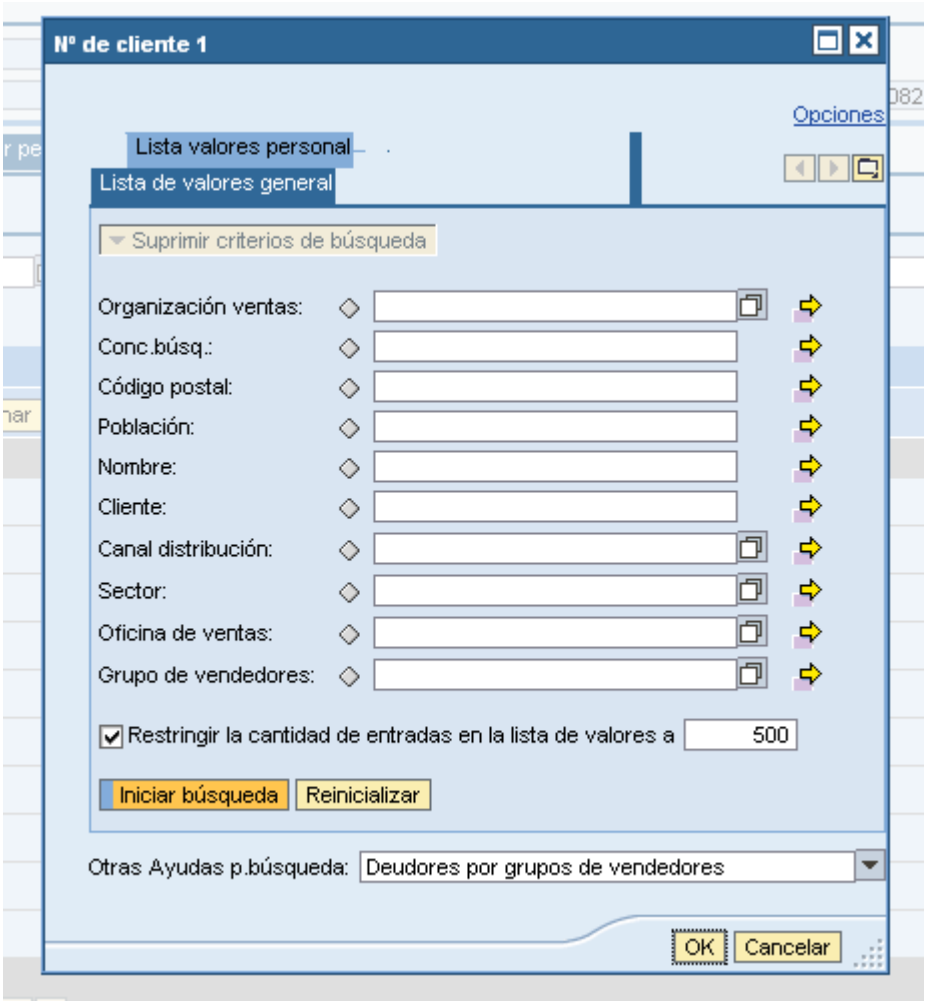
En este apartado el usuario podrá crear y consultar pedidos. Si el usuario es empleado deberá seleccionar un cliente al cual va a suplantar. Para seleccionar el cliente deberá

posicionarse sobre el campo identificador de cliente y pulsar el botón pequeño que tiene asociado el campo:



Pantalla 14

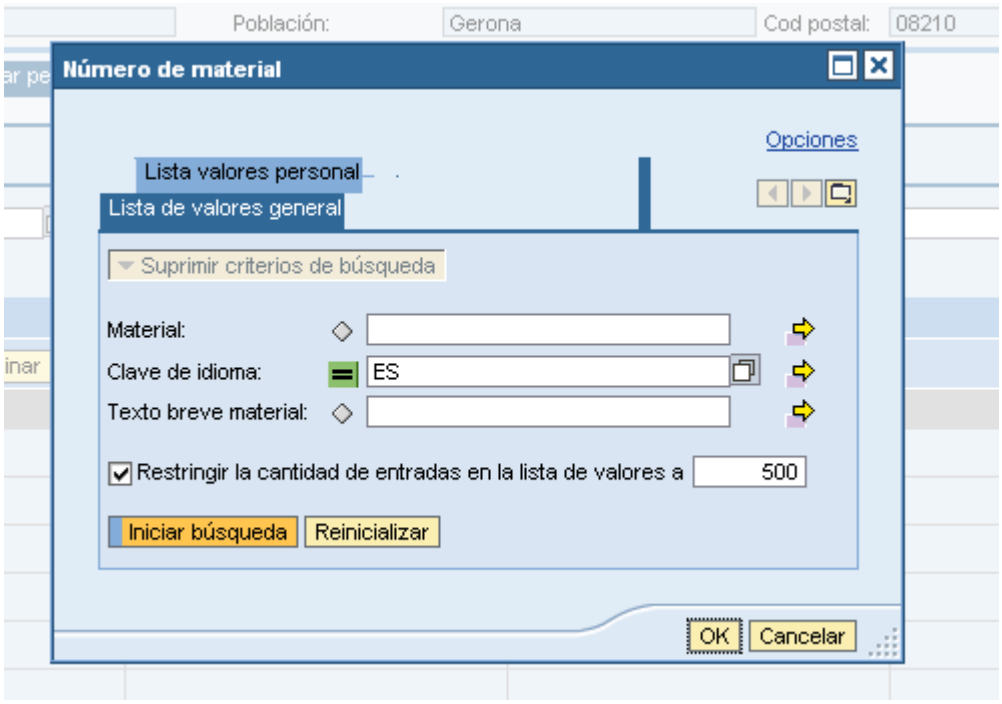
Una vez seleccionado se le abrirá una pequeña pantalla para poder ejecutar búsquedas de clientes. El empleado tendrá distintas formas de ejecutar la búsqueda del cliente:



Pantalla 15

En el caso del usuario cliente el sistema recupera su identificador y sus datos y lógicamente no puede seleccionar cliente.

Para realizar un pedido el proceso es el mismo para las dos tipologías de usuarios.
El primer paso es posicionarse sobre el campo material y pulsar el botón localizado al lado del campo. De este modo se abrirá una pantalla que permite ejecutar búsquedas de materiales:

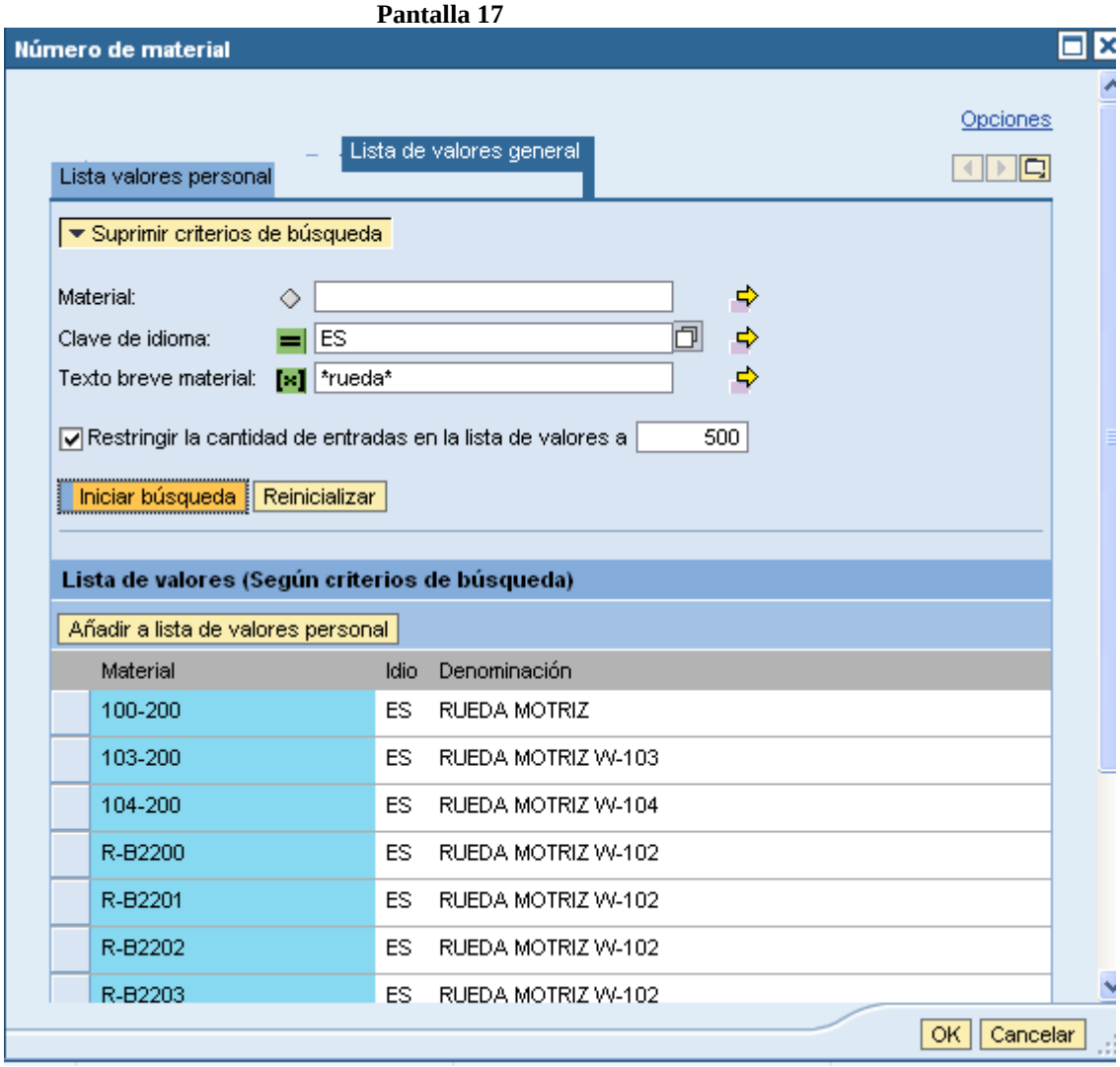


Pantalla 16

Se puede buscar un material por descripción y por código de material, admitiendo el "* " para completar secuencias y pulsar el botón "iniciar búsqueda"

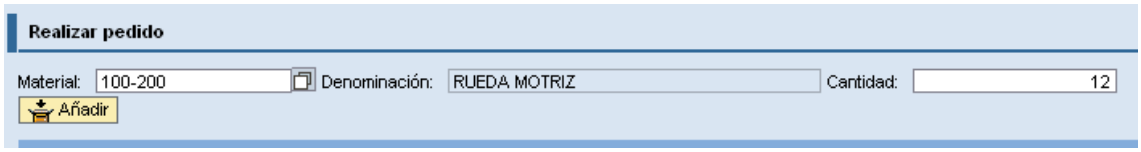

Población:
Gerona
Cod postal:
Número de material
Opciones
Lista valores personal
Lista de valores general
Suprimir criterios de búsqueda
Material:
Clave de idioma:
ES
Texto breve material:
rueda
Restringir la cantidad de entradas en la lista de valores a
500
Iniciar búsqueda
Reinicializar
OK
Cancelar

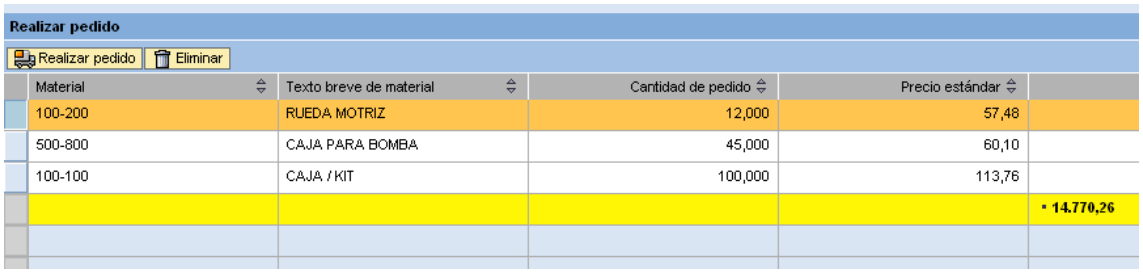
Pantalla 17


Número de material
Opciones
Lista valores personal
Lista de valores general
Suprimir criterios de búsqueda
Material:
Clave de idioma:
ES
Texto breve material:
rueda
Restringir la cantidad de entradas en la lista de valores a
500
Iniciar búsqueda
Reinicializar
Lista de valores (Según criterios de búsqueda)
Añadir a lista de valores personal
Material
Idio
Denominación
100-200 ES RUEDA MOTRIZ
103-200 ES RUEDA MOTRIZ W-103
104-200 ES RUEDA MOTRIZ W-104
R-B2200 ES RUEDA MOTRIZ W-102
R-B2201 ES RUEDA MOTRIZ W-102
R-B2202 ES RUEDA MOTRIZ W-102
R-B2203 ES RUEDA MOTRIZ W-102
OK
Cancelar

Pantalla 18

Para seleccionar el material posicionarse sobre él y seleccionarlo con un clic. Una vez se carga el material en el campo correspondiente informar la cantidad y pulsar el botón “Añadir”





Pantalla 19

Se puede visualizar el total por línea y el total del pedido y si se quiere eliminar un artículo, seleccionándolo mediante un clic y pulsar el botón “Eliminar "será suficiente. Para traspasar el pedido a SAP pulsar el botón “Realizar pedido” y el sistema retorna la información sobre el pedido que acaba de crear SAP.

Pantalla 20

En ese momento se puede consultar el pedido en la otra pestaña de la pantalla

Doc.venta	Material	Cantidad de pe	Texto breve de	Fe.entrega	Precio estánda	Precio estánda	Peso neto
						▪ 6.987,91	
▾ 12759						▪ 3.593,65	
▪	R-B2201	7,000	Rueda motriz W-10	11.05.2010	111,65	781,55	0,000
▪	M-05	10,000	Pantalla plana LE 50	11.05.2010	281,21	2.812,10	170,000
▾ 12794						▪ 3.394,26	
▪	100-200	12,000	Rueda motriz	02.06.2010	57,48	689,76	30,000
▪	500-800	45,000	Caja para bomba	02.06.2010	60,10	2.704,50	540,000

Pantalla 21

La información que se puede visualizar es la siguiente:

- Cantidad
- Código de material y descripción
- Fecha de entrega
- Precio unitario
- Precio unitario * la cantidad
- Precio total del pedido
- El volumen del material
- El total de todos los pedidos que hay pendientes de entrega
- El estado del artículo: si esta en stock o no.

10. Bibliografía

[1] SDN, SAP Developer Network (SDN): Downloads, Discussions, eLearning, and Documentation for Developers

[3] UML, http://www.uml.org/

[4] UML, http://www.omg.org/

[5] SAP, http://www.sap.com/index.epx

[6] SDN alv http://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/library/uuid/d0b6a153-a132-2d10-bcb3-ac585332542a

[7] Help SAP, http://help.sap.com/

[8] SAP AG, http://en.wikipedia.org/wiki/SAP_AG

[9] SAP, http://sap.ittoolbox.com/

[10] Licencias, https://websmp201.sap-ag.de/licenseauditing

Manuales consultados:

Manual NET310 ABAP Web Dynpro

Manual SCM600 Sales and Distribution Processes

Manual SCM520 Compras mySAP Supply Chain Management

Manual SCM510 Gestión de stocks e Inventario mySAP Supply Chain *Management*

Manual SCM500Procesos en el aprovisionamiento mySAP Supply Chain Management

Manual Web Infrastructure Concepts for SAP Web Application Server

Manual SAPTEC Bases de la solución mySAP Technology mySAP Technology

Libros consultados:

Especificació de sistemes orientats a objectes amb notació UML EDICIONS UPC.